VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INFORMAČNÍ SYSTÉM PRO SWINGOVOU TANEČNÍ ŠKOLU

INFORMATION SYSTEM FOR A SWING DANCE SCHOOL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ ZAVADIL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Zavadil Jiří**
Program: Informační technologie
Název: **Informační systém pro swingovou taneční školu**
Information System for a Swing Dance School
Kategorie: Informační systémy

Zadání:

1. Seznamte se s principy tvorby webových aplikací, dostupnými prostředími a frameworky. Dále prostudujte API bankovních institucí pro potvrzování bezhotovostních plateb.
2. Analyzujte požadavky na informační systém swingové taneční školy zahrnující správu kurzů a registrace do nich, tvorbu kalendáře kurzů, vyvažování počtu studentů v kurzech, zveřejňování výukových videí a možnost zpracování plateb s využitím napojení na API banky.
3. Navrhněte informační systém dle požadavků, návrh konzultujte s vedoucím.
4. Implementujte navržené řešení a otestujte jeho funkčnost na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a diskutujte další možné pokračování v tomto projektu.

Literatura:

- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9.
- Welling, L., Thomson, L.: PHP a MySQL: Kompletní průvodce vývojáře. CPress, 2017.
- Fio Banka: API Bankovnictví (dokumentace). Dostupné na: https://www.fio.cz/bankovni-sluzby/api-bankovnictvi

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 21. října 2019

Abstrakt

Tato práce se zabývá návrhem a následným vytvořením informačního a registračního systému pro swingovou taneční školu. Systém je vytvořen jako webová aplikace postavená na vývojové platformě Firebase. Pro implementaci prezentační části byl zvolen JavaScriptový framework Angular. Nejdůležitějšími funkcemi systému je zpracování registrací a plateb na taneční kurzy a automatická komunikace s klienty. Systém byl navržen, implementován a otestován.

Abstract

This thesis deals with designing and creating an information and registration system for a swing dance school. The system is created as a web application based on Firebase development platform. The JavaScript framework Angular was chosen for implementation of the presentation part. The most important functions of the system are processing of registrations and payments for dance courses and automatic communication with clients. The system was designed, implemented and tested.

Klíčová slova

informační systém, registrační systém, swing, Angular, Firebase, webová aplikace, NoSQL, AngularFire, Google Cloud Platform, tanec

Keywords

information system, registration system, swing, Angular, Firebase, web application, NoSQL, AngularFire, Google Cloud Platform, dance

Citace

ZAVADIL, Jiří. *Informační systém pro swingovou taneční školu.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém pro swingovou taneční školu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jiří Zavadil
29. července 2020

Poděkování

Tímto chci poděkovat svému vedoucímu panu Ing. Vladimíru Bartíkovi Ph.D. za vedení mojí bakalářské práce.

Obsah

Kapitola 1

Úvod

Poslední dekáda zaznamenala napříč nejen Evropou návrat obliby swingu jako společenského tanečního stylu. Postupně vznikala a vznikají nová taneční studia úzce specializovaná na tento styl hudby. Spolu se vzrůstající základnou swingových tanečníků začal růst i počet různých swingových festivalů a workshopů

Vznikající taneční studia mají na počátku podobu malých skupin nadšenců, kteří se snaží na daném místě rozšiřovat komunitu tanečníků pomocí pořádání kurzů. Ze skupin nadšenců postupně vznikaly a vznikají profesionálně organizované firmy typu tanečních studií. Při růstu řeší otázku profesionalizace a zefektivňování provozu skrze nastavování různých firmeních procesů. Jednou takovou je i spolupracující taneční škola, na základě jejíž potřeb bylo sestaveno zadání pro tuto práci.

Samotné udržování nastavených procesů je nevýdělečná exekutiva navíc, je tu tedy prostor pro podporu vhodnými softwarovými nástroji. Takových použitelných nástrojů je dnes k dispozici široká škála (registrační, docházkové, účetní systémy apod.). Problém je však vybrat tu správnou skupinu nástrojů. Ne všechny nástroje se totiž hodí pro podporu provozu swingové taneční školy. Při implementaci těchto nástrojů časem nastává také problém integrace a propojení jednotlivých nástrojů mezi sebou.

Klíčovými procesy pro firmu typu taneční školy je registrace nových klientů do tanečních kurzů a retence stávajících klientů. Retence - schopnost udržet si stávající klienty do dalších pokračování kurzů je zvláště klíčová, neboť je známé, že akvizice nových klientů je z hlediska marketingových nákladů řádově nákladnější než udržení stávajících.

Taneční školy dnes mají možnost tuto situaci řešit implementací široké škály různých registračních a docházkových systému. Většinou se však jedná o obecná řešení, která nejsou uzpůsobená přímo potřebám swingové taneční školy.

Dobré řešení na míru potřebám rostoucí swingové taneční školy, které by podporovalo nejen proces registrace ale celkově chod swingové taneční školy zatím chybí.

Proto jsme se ve spolupráci s taneční školou rozhodli takový systém začít tvořit. Swingových tanečních škol v posledních letech přibylo napříč evropou větší množství. Proto věříme, že produkt tohoto typu najde i širší uplatnění. V ideálním případě by měl být v budoucnu poskytován jako SAAS[1].

Z diskuzí se spolupracující taneční školou se ukázala jako primární potřeba vyřešit efektivně specifický proces registrace kurzistů na taneční kurzy. Cílem práce tedy bude co největší automatizace tohoto procesu včetně nutné komunikace, která nyní silně časově zatě-

[1] Software as a service (SAAS) – software jako služba

žuje organizátory lekcí. Řešení bude v prvním kroku stavěno pro využití jednou organizací. V budoucnu by poté mohlo dojít k jeho zobecnění do podoby software jako služby.

Stanovený rozsah je dobře představitelný a ohraničený, proto jsme při vývoji postupovali klasickou metodou vývoje Waterfall. Výstup práce bude sloužit jako základ pro další vývoj, jehož priority budou již méně predikovatelné. V potenciálních dalších fázích se tak počítá s agilním způsobem vývoje.

Kapitola 2

Analýza požadavků

Swingové taneční školy jsou postaveny na pořádání dlouhodobých tanečních kurzů či jednorázových workshopů. Dlouhodobé kurzy jsou primární z hlediska výše příjmů. Kurzy jsou tak organizovány v rámci semestrů (zpravidla dva až čtyři za rok), probíhají v tanečních sálech, kde výuku vedou taneční lektoři.

Kurzy jsou sestavené zpravidla z devíti lekcí konaných jednou týdně. Počet a frekvence se však může měnit. Do kurzů se hlásí studenti jednotlivě či v páru v tanečních rolích jako leader či follower. Kapacita kurzů bývá většinou omezena na třicet účastníků. Důležitá je vyváženost kurzu. Během období registrací je snahou dosáhnout co největšího vyvážení rolí leader a follower.

Výuku na kurzu vedou v případě párových tanců dva lektoři, v případě sólo tanců lektor jeden. Různé kurzy se mohou odehrávat na různých místech (tanečních sálech). V průběhu semestru se však místo daného kurzu již nemění.

Uvažovaný systém bude využíván během celého tanečního semestru. Jeho role se však bude v průběhu měnit. V období registrací, které trvá obvykle několik týdnů na počátku každého tanečního semestru, bude plnit roli především jako registrační systém. Během tanečního semestru pak bude sloužit i dalším účelům podporující chod taneční školy.

V systému taneční školy se tedy vyskytují tyto entity:

- **Kurz** - sestavený z několika lekcí
- **Lekce** - jedno konání kurzu
- **Semestr** - obsahuje kurzy
- **Lektor** - vede lekce
- **Student** - registruje se na kurzy a navštěvuje jejich lekce
- **Lokace tanečního sálu** - místo odehrávání lekcí

2.1 Funkční požadavky

Zde popisuji funkční požadavky na systém dle priorit spolupracující taneční školy:

- **Proces registrace kurzistů:** Příjem objednávek a zpracování platby za kurzy je pro taneční školu klíčový. Aby celý proces mohl fungovat, je potřeba realizovat následující dílčí části:

- **Kalendář vypsaných kurzů:** Modul kalendáře umístitelný do webové stránky taneční školy bude sloužit jako týdenní přehled dostupných kurzů. Kurzisté si zde budou moci vybrat kurz dle zájmu a svých časových možností a zaregistrovat se do něj.
- **Detail kurzu:** Zájemci o kurzy zde získají podrobnější informace o konaném kurzu. Popis obsahu kurzu, představení lektorů, informace o ceně, obsazenosti a možných slevách.
- **Registrace do kurzu:** Kurzy swingu mají obvykle několik specifik. Kurzisté se mohou registrovat v páru ale i jednotlivě v rolích obecně jako leader či follower (nemusí korespondovat s pohlavím). Swing je společenský tanec postavený na párové improvizaci, proto je na kurzech pravidlem takzvaného točení, kdy se leadeři a followeři přibližně po 5 minutách střídají dokola. Tento způsob vedení lekce umožňuje registrace jednotlivců, kteří nemají partnera. Registrace v párech však bývají zvýhodňovány různými slevami.
- **Vyváženost rolí na kurzu:** Možnost hlásit se jako jednotlivec přináší benefit otevřenosti vůči většímu množství zájemců. Na druhou stranu v období registrací do kurzů nastávají problémy s vyvažováním jednotlivých rolí. Vyváženost je z hlediska účastníků důležitým parametrem kvality kurzu. Organizačně je jí však v průběhu období registrací náročné dosáhnout. Systém by toto měl činit automatizovaně a ulehčit tak opakující se exekutivní zátěži. Je žádoucí aby systém umožnil nastavitelnou míru nevyváženosti na kurzu. Dnes je tento problém řešen pomocí čekací listiny.
- **Automatizace komunikace se zájemcem:** Během období registrací je silně zvýšená potřeba komunikace s novými či stávajícími klienty. Tato komunikace by se dala rozdělit do dvou kategorií: procesní a péče o zákazníky. Kvůli zahlcujícímu množství procesní komunikace v období registrací zbývá menší prostor pro individuální komunikaci pečující o zákazníky, která je však v tomto období klíčová z hlediska získávání a udržení klientů. Systém by tak měl ideálně vyřešit veškerou procesní komunikaci, tak aby organizátorům uvolnil ruce pro individuální komunikaci. Systém by měl provést nového kurzistu celým procesem registrace a v průběhu ho informovat o všech nezbytných věcech. Procesní komunikace se skládá z následujících kroků:
  * informace o přijetí registrace
  * informace o schválení registrace
  * informace o zařazení zájemce na čekací listinu
  * informace o změně stavu registrace v rámci čekací listiny
  * zaslání platebních údajů
  * informace o přijaté platbě a zařazení do kurzu
  * upozornění o nezaplacení
  * informace vyřazení z registrací při nezaplacení

- **Správa kurzů, semestrů, lektorů, lekcí, studií:** Administrátor systému bude potřebovat možnost snadné správy a editace dílčích entit systému. Jedná se o především o možnost vytvářet, editovat a odstraňovat tyto entity. Dále pak nastavovat parametry celého systému jako je například obsah procesní komunikace s klientem, sleva při párové registraci či studentská sleva, míra nevyváženosti kurzů a případně další.

- **Správa a řízení stavu registrace, plateb a objednávek:** Tato sada funkcionalit je nezbytná pro účely účetnictví.
- **Záznam docházky studentů:** Z průzkumů, které prováděla spolupracující taneční škola vyplynulo, že pro to, zdali se kurzista stane dlouhodobým swingařem či nikoliv, jsou klíčové úvodní dva kurzy. Udržení klientů během tohoto období je pro školu zásadní. Často se například stává, že kurzista jednou vynechá kurz a příště má obavu, že už další lekce nebude zvládat, tak raději přestane chodit úplně. Ukázalo se ale, že v takovém případě stačí jen malé cílené povzbuzení formou krátké zprávy a kurzista obavy překoná a v kurzu pokračuje. Vedení docházky je proto důležitým zdrojem dat pro jednoduchou predikci odchodu zákazníka. Součástí docházkového systému by pak mělo být i vedení výkazu práce lektora pro účely vyplácení mezd.
- **Záznam videa z lekce:** Běžnou praxí na swingových lekcích a workshopech je to, že lektoři na konci lekce rekapitulují výuku - trénované taneční kroky a figury. Kurzisti si přitom mohou lektory natáčet a videa využít pro své účely tréninku a přípravy na další lekci. Systém by tuto praxi měl podporovat a umožnit jednomu snadno rekapitulaci natočit a sdílet ji ostatním v rámci profilu kurzistů.

2.2 Nefunkční požadavky

Následuje výpis nefunkčních požadavků:

- **Dostupnost systému:** Systém bude využíván vždy v průběhu celého tanečního semestru. Měl by být dostupný online 24 hodin, 7 dní v týdnu.
- **Responzivita:** Je velice pravděpodobné, že uživatelé budou interagovat se systémem prostřednictvím mobilních zařízení, je tedy nutné zajistit responzivitu celého systému.
- **Flexibilní:** bez nutnosti instalace - Aplikace by měla být dostupná bez nutnosti instalace ve formě webové aplikace.

2.3 Analýza případů užití

Uživatele v systému lze rozdělit do několika rolí:

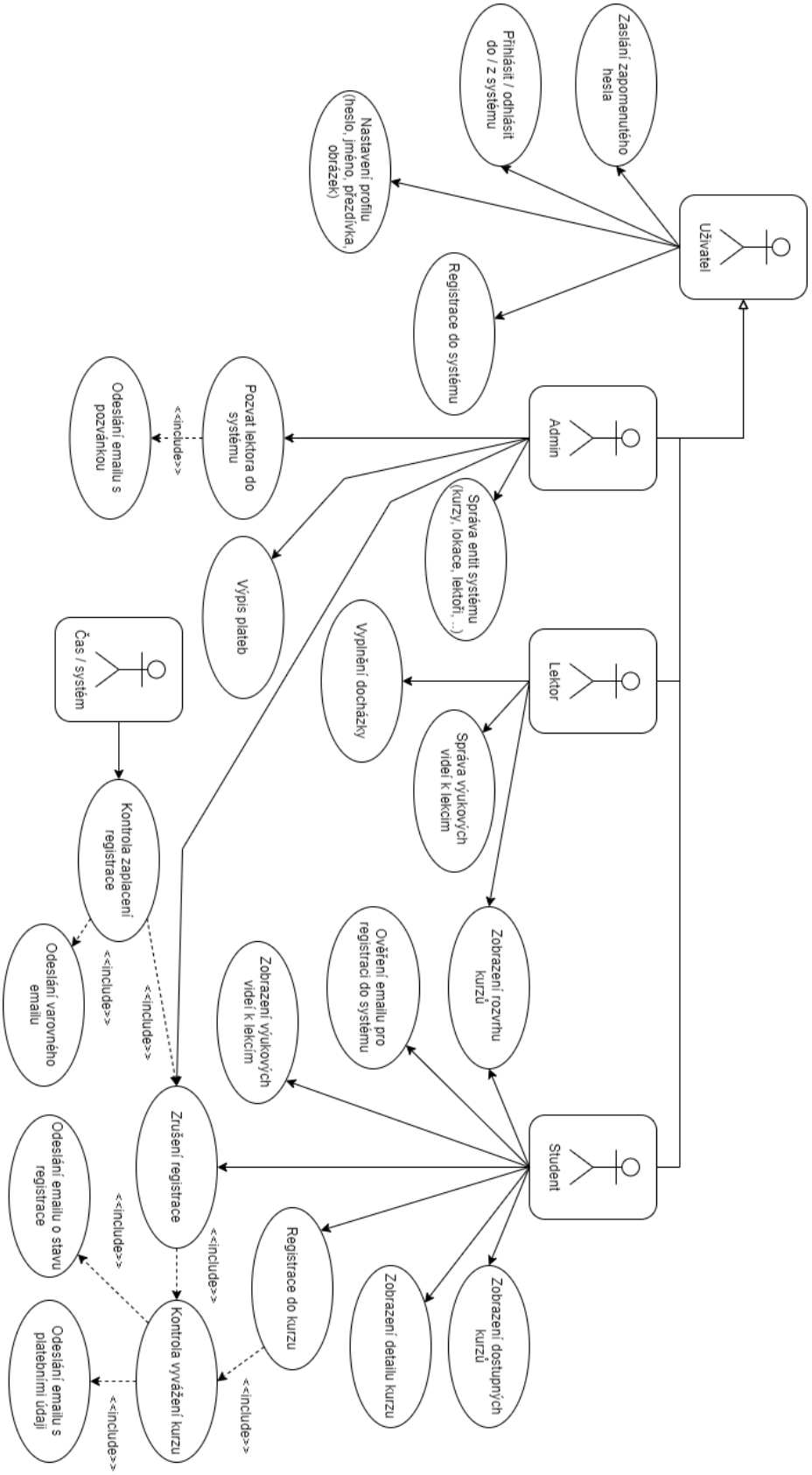
- **Administrátor:** Administrátor je uživatel s právem spravovat nastavení systému a jednotlivé entity (kurzy, semestry, lokace, lektory, objednávky).
- **Lektor:** Lektor vede lekce a zaznamenává na nich docházku. Má přístup ke kalendáři svých kurzů a po každé lekci může do systému přidat video z odučené lekce.
- **Student:** Je uživatel s možnostmi zobrazit dostupné kurzy a jejich detailní popisy, registrovat se do kurzů, spravovat své objednávky, zobrazit videa z lekcí.

Entita Uživatel je pak nadtřídou všech ostatních rolí systému. Zahrnuje případy užití, které mají všechny role společné. Jedná se o:

- Registraci do systému
- Nastavení profilu

- Přihlášení / odhlášení ze systému
- Zaslání zapomenutého hesla

Jednotlivé uživatelské role a jejich případy užití shrnuje přehledně obrázek 2.1.

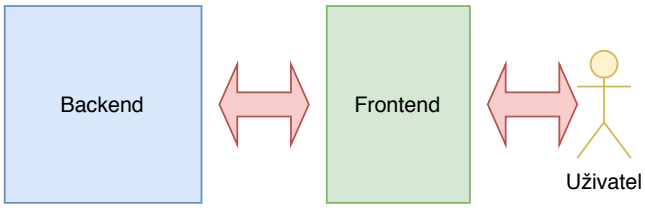


Obrázek 2.1: Diagram případů užití.

Kapitola 3

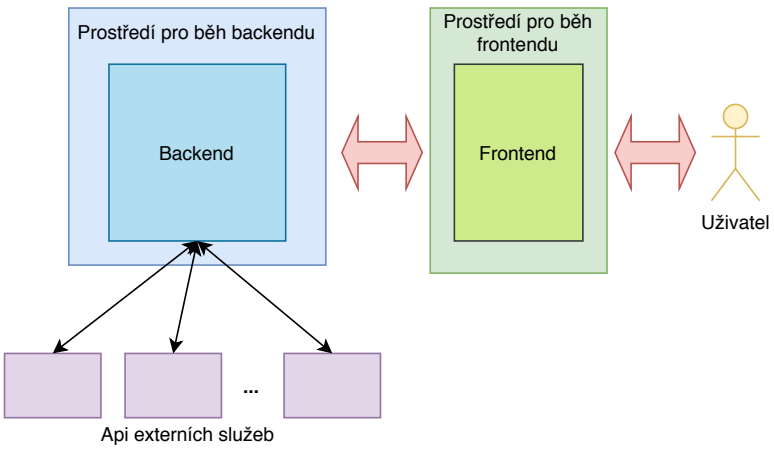
Výběr technologií

V rámci této práce vytvářím webovou aplikaci - software. Z hlediska architektury lze pohlížet na software tak, že se skládá ze dvou částí: prezentační část (dále jen frontend), se kterou přichází do kontaktu uživatel, a výpočetní část (dále jen backend), která realizuje obsloužení požadavků uživatele vizte obrázek 3.1.



Obrázek 3.1: Pohled na software z hlediska architektury.

Backend a frontend systému budou fungovat v různých prostředích vizte obrázek 3.2. V případě frontendu uvažujeme tímto prostředím moderní webové prohlížeče. V případě prostředí pro běh backendu se bude jednat o nějaký server či serverovou službu. Předpokladem je i to, že backend bude využívat užitečné externí služby.



Obrázek 3.2: Architektura webové aplikace.

Výběr technologií budeme tedy řešit celkem z pěti hledisek:

- Prostředí pro nasazení backendu
- Technologie pro frontend
- Technologie pro backend
- Technologie pro komunikaci frontend - backend
- Podpůrné technologie - externí služby a jejich API[1] a další nástroje

3.1 Technologie pro Frontend

Z [13] vyplývá, že aktuálně nejoblíbenějším programovacím jazykem pro webové prohlížeče je Javascript. Při vývoji webových aplikací pomocí javascriptu se dnes obvykle využívají různé specializované JavaScript frameworky (dále JSF). JSF poskytují předpřipravenou funkcionalitu čímž výrazně usnadňují práci, což vede na urychlení vývoje aplikace. V dnešní době je dostupná celá řada různých JSF. Zásadní je vhodný výběr. Samotná otázka stanovení vhodných kritérií pro výběr JSF může být obtížná. Proto při svém výběru vycházím z výstupů studie[18] mezi zkušenými vývojáři, která se na tuto otázku zaměřovala. Z výstupů vyplývají tři hlavní kritéria:

- Adekvátnost dokumentace
- Velikost a účast komunity
- Pragmatika JSF

Dalšími důležitými kritérii pro výběr vhodného JSF jsou frekvence aktualizací a určitá dospělost(stáří) a historie vývoje JSF. V kontextu účelu této práce přidávám ke zmíněným kritériím i kritérium vlastní zkušenosti s danou technologií.

3.1.1 Dostupné javascriptové frameworky

Na základě analýzy[15] bylo zjištěno, že pouze třem známým JSF roste popularita, zatímco ostatní spíše co se využívání týče stagnují:

- Angular
- React
- Vue.js

Následuje shrnutí zásadních výhod a nevýhod[15].

[1] Application Programming Interface (API) – rozhraní pro programování aplikací

Angular

Výhody:

- Aplikace běží rychleji díky menšímu přenosu dat a neaktualizování celé stránky, ve chvíli kdy uživatel prochází aplikací.
- Díky obousměrné datové vazbě dochází v Angularu k předání změn v modelu okamžitě do prezentační vrstvy.
- MVVM[2] umožňuje pracovat se stejnou kolekcí dat samostatně v rámci jedné aplikace.
- Struktura a architektura frameworku je postavená speciálně pro lepší škálovatelnost aplikace.
- Modularita.

Nevýhody:

- Angular je náročnější na prvotní naučení, jelikož má v porovnání s frameworky React a Vue.js (které mají jednu základní strukturu), základních struktur několik.
- Časté aktualizace mohou působit potíže při adaptaci aplikace na provedené změny.

React

Výhody:

- Používá virtuální DOM[3], díky čemuž dochází k efektivní aktualizaci prezentační vrstvy aplikace.
- Vykreslování aplikací na straně serveru.
- Relativně lepší SEO[4] v porovnání s frameworky Angular a Vue.js.

Nevýhody:

- Nutnost importování stavových a modelových knihoven, jelikož React neimplementuje architekturu MVC[5].
- Kvalita dokumentace.

Vue.js

Výhody:

- Malá velikost, rychlost.
- Efektivní zpracování obousměrného dynamického provázání dat.
- Lze se poměrně rychle naučit.

[2] Model-view-viewmodel (MVVM) – softwarová architektura
[3] Document Object Model (DOM) – objektově orientovaná reprezentace dokumentu
[4] Search engine eptimization (SEO) – optimalizace pro vyhledávače
[5] Model-view-controller (MVC) – softwarová architektura

Nevýhody:

- Oproti ostatním frameworkům má menší komunitu vývojářů.

Tabulka 3.1: Subjektivní model hodnocení frameworků (hodnocené 1 až 5 (vyšší znamená lepší))

	Dokumentace a komunita	Pragmatika	Aktualizace	Dospělost (stáří)	Vlastní zkušenost	Celkem
Angular	4	3	4	4	4	19
React	3	3	3	3	2	14
Vue.js	3	3	3	2	1	12

Na základě tabulky 3.1 hodnocení JSF jsem pro frontendovou technologii zvolil Angular ve verzi 8, která je aktuální v době psaní této práce.

3.1.2 Angular

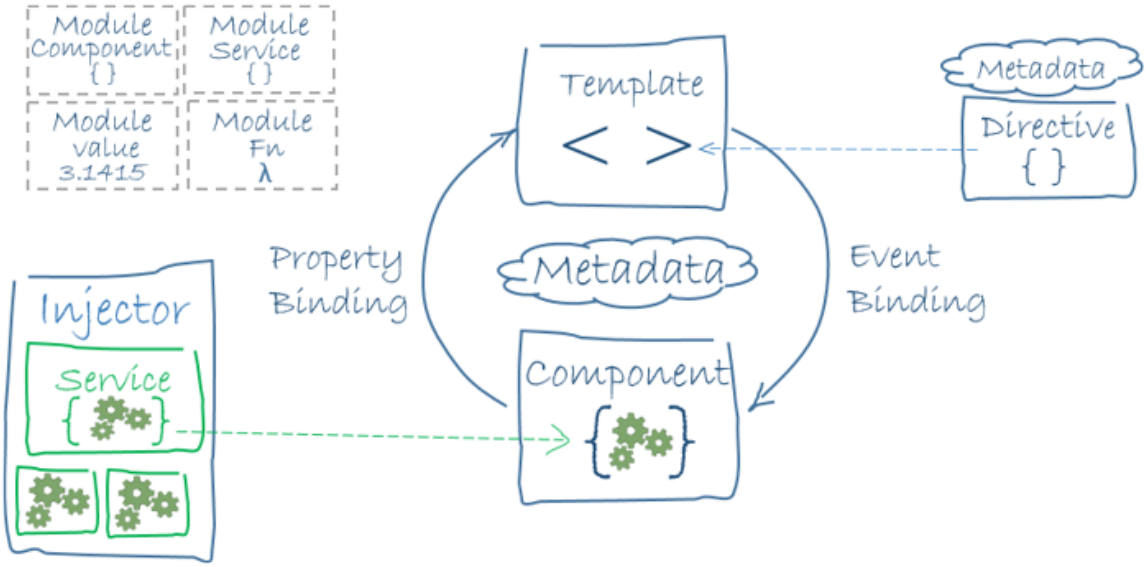
Angular je framework pro tvorbu SPA[6]. Využívá značkovací jazyk HTML[7] a skriptovací jazyk TypeScript. Architektura aplikace vytvořené v Angularu se opírá o několik základních konceptů. Základním stavebním blokem jsou moduly neboli takzvané NgModules. Moduly v Angularu zaštiťují kód aplikace do funkčních sad, které spolu vzájemně souvisejí. Aplikace vytvořená v Angularu je pak tvořena sadou modulů. Aplikace má vždy alespoň jeden kořenový modul, který umožňuje zavádění systému a typicky obsahuje více dalších funkčních modulů. Dalšími stavebními bloky jsou komponenty, které definují takzvané pohledy (views). Komponenty jsou elementy viditelné na obrazovce, které Angular spravuje na základě logiky programu. Vizuální podoba komponent je definována pomocí HTML a CSS[8]. Chování komponenty se pak definuje pomocí TypeScriptu.

Moduly, komponenty a služby, jsou realizovány pomocí konstruktů tříd a dekorátorů v jazyce TypeScript. Dekorátory slouží pro označení typu a poskytují metadata, které dávají Angularu informaci jak daný modul, komponentu či službu používat. Komponenty aplikace typicky definují více pohledů. Angular poskytuje směrovací službu (router), která pomáhá definovat navigační cesty pro všechny pohledy.

Moduly (NgModules)

Moduly v Angularu deklarují kompilační kontext pro sadu komponent. Každá aplikace má kořenový modul, běžně pojmenovaný AppModule, který poskytuje zaváděcí mechanismus, který zahájí běh aplikace. Aplikace obvykle obsahuje několik modulů. Moduly mohou importovat funkcionality z ostatních modulů a naopak mohou být importovány do jiných modulů. Organizace kódu do modulů pomáhá dosáhnout lepšího pořádku v celém projektu, lepší znovupoužitelnosti kódu. Moduly navíc umožňují využití techniky takzvaného líného načítání ("lazy-loading"). Díky této technice lze načítat moduly až ve chvíli, když jsou potřeba, čímž se minimalizuje doba pro počáteční načtení aplikace.

[6]Single page aplication (SPA) – jednostránková aplikace
[7]Hypertext Markup Language (HTML) – značkovací jazyk pro tvorbu webových stránek
[8]Cascading Style Sheets (CSS) – jazyk pro popis způsobů zobrazení elementů na stránkách



Obrázek 3.3: Architektura angular aplikace. (Převzato z [12])

Komponenty (components)

Každá aplikace má alespoň jednu kořenovou komponentu, která spojuje celou hierarchii komponent s objektovým modelem dokumentu (DOM). Každá komponenta definuje třídu, která obsahuje aplikační data a logiku. Komponenty jsou spojeny s HTML šablonou, která definuje pohled. To, že se jedná o komponentu určuje dekorátor @Component(), který poskytuje šablonu a další metadata specifické pro komponentu.

Šablony (templates) a vázání dat (data binding)

Pomocí šablon se v Angularu implementuje vizuální podoba komponent. Šablony jsou realizovány pomocí kombinace značkovacího jazyka HTML a značkami Angularu. Angular podporuje obousměrné vázání dat. To znamená, že změny v DOM, které mohou být způsobeny například uživatelem, jsou reflektovány v programových datech. Naopak změny programových dat se promítnou v DOM.

Služby (services)

Služby jsou pro data, nebo programovou logiku, které nejsou přímo spojeny se specifickým pohledem a u kterých se chce, aby byly sdíleny více komponentami. Definici služby předchází dekorátor @Injectable().

Směrování (routing)

Směrovací modul Angularu nabízí definování navigačních cest mezi různými stavy aplikace. Tento modul je navržen podle známých navigačních konvencí prohlížečů:

- Vložení URL do adresního řádku prohlížeče nasměruje aplikaci na odpovídající stránku.
- Kliknutí na odkaz zapříčiní přechod na novou stránku.

- Kliknutí na prohlížečová tlačítka zpět případně vpřed způsobí přechod na předchozí, případně následující stránku.

3.2 Prostředí pro nasazení backendu

Při výběru prostředí pro nasazení backendu dnes můžeme uvažovat vícero způsobů řešení, které lze roztřídit dle míry abstrakce do následujících kategorií:

- Prostředí vlastního serveru
- Prostředí poskytnuté službou typu IAAS[9]
- Prostředí poskytnuté službou typu PAAS[10]
- Prostředí poskytnuté službou typu BAAS[11]

3.2.1 Prostředí vlastního serveru

Prostředí vlastního serveru přináší naprostou volnost při konfiguraci, což je někdy možná výhoda. Pro naše účely je to však spíše nevýhoda. Konfigurace takového prostředí je časově náročná a vyžaduje mnoho zkušeností. Zásadní nevýhodou je pak i obtížná škálovatelnost výpočetního výkonu a veliká vstupní investice.

3.2.2 Prostředí IAAS

Prostředí typu IAAS přináší oproti vlastnímu serverovému řešení výhodu odlehčení vstupních nákladů a velikou škálovatelnost výkonu. Hardware je předkonfigurovaný, což šetří významně čas. Je zde volnost vlastní konfigurace operačního systému či nasazení jakéhokoliv spustitelného softwaru. Pro naše účely je však nutnost instalace a konfigurace OS a SW nyní také spíše nevýhoda, jelikož vyžaduje zkušenosti a je časově náročná [19].

3.2.3 Prostředí PAAS

Prostředí typu PAAS oproti IAAS odstraňuje nutnost instalace a konfigurace OS. Umožňuje běh pouze podporovaných aplikací. Může poskytovat nakonfigurovaný a aktualizovaný middleware (například databáze, aplikační servery, web servery, a další) či podpůrné nástroje pro monitoring [19].

3.2.4 Prostředí BAAS

Backend (pro naše účely přesnější pojmenování MBASS[12]) jako služba snižuje nutnost vývoje a správy aplikačního backendu. Oproti PAAS přináší předpřipravené funkcionality jako jsou například cloudová úložiště, správa databází, realtime databáze, push notifikace atp. Nevýhodou oproti PAAS je menší míra flexibility, větší svázanost [3].

[9] Infrastructure as a service (IAAS) – infrastruktura jako služba
[10] Platform as a service (PAAS) – platforma poskytnutá jako služba
[11] Backend as a service (BAAS) – backend jako služba
[12] Mobile backend as a service (MBAAS) – mobilní backend jako služba

3.2.5 Zhodnocení a výběr prostředí

Každé z výše zmíněných řešení má své popsané výhody a nevýhody. Obecně však lze říci, že s přibývající mírou abstrakce klesá náročnost správy daného prostředí. Na druhou stranu však klesá i míra flexibility prostředí a obvykle se zvyšuje i cena za poskytovanou službu. Pro účely mojí práce jsem se rozhodl využít prostředí typu BAAS. Hlavní důvod výběru je ten, že díky předpřipravené funkcionalitě, toto prostředí umožní rychleji dodat potřebnou funkcionalitu a zaměřit se tak více na to, co je v počáteční fázi projektu potřeba. Tím je uživatelská přívětivost systému, stabilita a rychlé dodání. Tyto benefity výrazně překonají vyšší náklady spojené s využíváním služeb typu BAAS a menší flexibility a možnosti konfigurace takových prostředí.

3.2.6 Analýza dostupných BAAS služeb

Z provedené rešerše dostupných (M)BAAS služeb jsem předvybral tři nejpoužívanější. Jedná se o Firebase od Googlu, Azure App service od společnosti Microsoft a AWS Amplify od Amazonu. Všechny tři služby srovnávám subjektivním hodnocením dle zvolených kritérií v tabulce 3.2.

Tabulka 3.2: Subjektivní model hodnocení technologií (hodnocené 1 až 10 (vyšší znamená lepší))

	Jednoduchost použití	Zabezpečení	Dostupné funcionality	Cenový model	Vlastní zkušenost	Celkem
Azure APP Service	7	10	8	9	1	35
AWS Amplify	8	10	8	9	1	36
Firebase	8	10	8	9	4	39

Při hodnocení vybraných služeb v kontextu potřeb naší aplikace vycházeli co do kvality a rozsahu služeb všechny řešení srovnatelně. Rozhodujícím bodem tedy pro mé rozhodování ve výsledku byla vlastní předchozí zkušenost. Pro další práci jsem tedy vybral službu Firebase od společnosti Google.

3.2.7 Firebase

Firebase je služba typu BAAS, která je součástí Google Cloud platformy. Firebase poskytuje řadu předpřipravených funkcionalit, které zajišťují většinu funkcí potřebných pro vývoj a běh webových a mobilních aplikací [14]. Pro kontext této práce jsou zásadní hlavně tyto podslužby:

- Cloud Firestore
- Cloud Functions
- Cloud Storage
- Authentication
- Hosting

Cloud Firestore

Cloud Firestore je v cloudu hostovaná, NoSQL[13] databáze ke které mohou aplikace přistupovat přímo přes nativní SDK[14]. Data jsou zde ukládána v rámci takzvaných dokumentů, které jsou organizovány do takzvaných kolekcí. Dokumenty mohou obsahovat subkolekce. Nad (sub)kolekcemi pak lze vytvářet různé CRUD[15] dotazy. Dokumenty umožňují uložení různých datových typů, od jednoduchých řetězců až po komplexní vnořené objekty [5].

Klíčové vlastnosti [5]:

- **Flexibilita:** Datový model Firestore databáze podporuje flexibilní, hierarchické struktury dat.
- **Expresivní dotazování:** Firestore databáze podporuje dotazy pro získání konkrétních dokumentů, či získání všech dokumentů v kolekci, které odpovídají parametrům poskytnutého dotazu. Dotazy umožňují řetězení vyhledávacích filtrů a nastavení řazením výsledků. Všechny dotazy jsou ve výchozím stavu indexovány, což znamená, že rychlost dotazů je úměrná počtu výsledků, nikoli velikosti uložených dat.
- **Aktualizace v reálném čase:** Ve chvíli, kdy nastane změna v libovolném připojeném klientovi, Firestore databáze se postará o propagaci informace o změně do všech připojených klientů.
- **Offline podpora:** Data, která se aktivně využívají se ukládají do mezipaměti. Aplikace tedy může nad těmito daty provádět operace, i když je zařízení ve stavu offline. Ve chvíli opětovného připojení do stavu online, Firestore databáze všechny lokální změny synchronizuje se vzdálenými změnami a automaticky vyřeší případné konflikty.
- **Přístupné z klientských zařízení:** Databáze je díky připravenému SDK přístupná přímo z frontendu aplikace, což znamená, že není potřeba aplikační server. Zabezpečení dat se provádí pomocí bezpečnostních pravidel (Security Rules), které popisuji níže.

Cloud Functions

Cloud Functions pro Firebase je serverless framework, který automaticky spouští serverový kód na základě Firebase funkcí a HTTPS[16] požadavků. Firebase funkce reagují na změny vytížení automatickým škálováním výkonu, které je provedeno rychlou změnou počtu instancí virtuálních serverů, na kterých funkce běží. Každá funkce běží v izolaci, ve svém prostředí a se svým nastavením [6].

Callables

Takzvané callables jsou ve Firebase serverové funkce, obsahující funkční logiku složitějších operací, ale také jednoduchých operací jako zápis, mazání, aktualizace. Díky SDK mohou

[13] Not Only SQL (NoSQL) – databázový koncept využívající jiné prostředky než tradiční relační databáze
[14] Software development kit (SDK) – sady vývojových nástrojů
[15] Create Read Update Delete (CRUD) – operace vytvoření, čtení, aktualizace a smazání
[16] Hypertext Transfer Protocol Secure (HTTPS) – protokol pro zabezpečenou komunikaci v počítačové síti

být tyto funkce volány přímo z klienta. Callables oproti klasickým HTTP[17] funkcím navíc umí [4]:

- Automatické přidání autentizačního tokenu do hlavičky požadavku, pokud je token přítomen. To umožňuje jednoduché zabezpečení přístupu k těmto funkcím.
- Automatické deserializování těla požadavku a validaci autentizačních tokenů.

Cloud Storage

Cloud Storage pro Firebase je služba určená k ukládání souborů. Firebase Cloud Storage má k dispozici SDK, pomocí kterého lze nahrávat a stahovat soubory přímo z klientského zařízení. V případě slabého síťového připojení je klientské zařízení schopno opakovat operaci od bodu, kde skončilo. Cloud Storage ukládá soubory v takzvaném Google Cloud Storage bucketu, který jej zpřístupňuje přes Firebase a Google Cloud. To umožňuje nahrávání a stahování souborů z mobilních klientů přes Firebase SDK, a provádět zpracovávání na straně serveru (například filtrování obrázků nebo transkódování videa). Cloud Storage škáluje potřebný výkon automaticky [7].

Klíčové vlastnosti [7]:

- **Robustnost:** Provádí nahrávání a stahování bez ohledu na kvalitu síťového připojení. V případě přerušení, se operace později znovu spustí tam, kde došlo k přerušení.
- **Zabezpečení:** Firebase SDK pro Cloud Storage integruje služby zabezpečení Firebase Authentication a Firebase Security Rules, pomocí kterých lze řídit přístup k souborům.
- **Škálovatelnost:** Cloud Storage umožňuje škálovat výkonnost potřebnou od fáze prototypu aplikace až do fáze produkce pomocí stejné infrastruktury.

Firebase Authentication

Většina aplikací obvykle potřebuje alespoň pro část svých funkcionali znát identitu uživatele. Firebase Authentication poskytuje serverové služby, SDK, a připravené knihovny uživatelského rozhraní k procesu ověření uživatelů v aplikaci. Podporuje ověření pomocí emailu a hesla, telefonního čísla, nebo ověření skrze přihlášení k účtům Google, Facebook, Twitter a další.

Díky tomu se vývojáři nemusí starat o autentizaci a nutná zabezpečení s tím spojená a mohou věnovat více času vývoji vlastní aplikace. Firebase Authentication se úzce integruje s ostatními Firebase službami a využívá průmyslové standardy jako OAuth 2.0 a OpenID Connect.

K přihlášení do systému je z pohledu vývojáře aplikace nutné nejdříve získat od uživatele přihlašovací údaje. Tyto přihlašovací údaje mohou být v podobě emailu a hesla, nebo OAuth tokenu. Poté se tyto údaje předají Firebase Authentication SDK. Server pak tyto údaje ověří a vrátí výsledek do klientské aplikace. Po úspěšném přihlášení jsou zpřístupněny základní informace o profilu uživatele [9].

[17] Hypertext Transfer Protocol (HTTP) – protokol pro komunikaci s webovými servery

Firebase Hosting

Služba Firebase Hosting slouží k hostování webového obsahu. Služba usnadňuje poskytování služeb a infrastruktury potřebné k nasazení škálovatelných, globálně dostupných webových aplikací. Infrastruktura dokáže obsloužit miliardy požadavků denně. Soubory jsou doručeny pomocí zabezpečeného protokolu HTTPS a SSL[18] přes CDN[19] [10].

Klíčové vlastnosti [10]:

- **Zabezpečení:** Poskytování obsahu přes zabezpečené připojení.
- **Statický i dynamický obsah:** Firebase Hosting podporuje hostování statického obsahu od CSS a HTML souborů, až k Express.js mikroslužbám nebo API.
- **Rychlé dodání obsahu:** Nahrané soubory jsou ukládány na rychlých SSD[20] discích v rámci CDN po celém světě. Rychlost doručení obsahu není závislá na místě kde se uživatel nachází.
- **Řízení změn:** Firebase CLI[21] umožňuje rychlé nasazení nových změn resp. verzí, či návrat k verzím předcházejícím.

Firebase Custom Claims

Firebase Admin SDK podporuje definování vlastních atributů v rámci uživatelských účtů. To umožňuje implementovat různé typy řízení přístupů, včetně řízení přístupu založené na rolích. Tyto vlastní atributy dávají uživatelům různé úrovně přístupu, které jsou vynuceny bezpečnostními pravidly aplikace [8].

Následující kód 3.4 demonstruje přidělení takzvaného admin Custom Claim k uživateli odpovídajícímu identifikátorem *userId*.

```
admin.auth().setCustomUserClaims(userId, {admin: true}).then(() => {
    // prideleni Custom Claimu k uzivateli probehlo uspesne
    // zmeny se projevi pri dalsim generovani uzivatelskeho tokenu
    // - typicky po dalsim prihlaseni
});
```

Obrázek 3.4: Ukázka přidělení takzvaného admin Custom Claim k uživateli s identifikátorem *userId*

Firebase Security Rules

Takzvané Firebase security rules neboli bezpečnostní pravidla stojí mezi daty a škodlivými uživateli. Jsou to pravidla, která udávají kdo může provádět čtecí nebo zapisovací operace nad kterými dokumenty. Pravidla fungují porovnáváním vzoru s cestou, ke které se přistupuje a potom aplikování vlastních podmínek k povolení přístupu k datům [11].

Pro databázi Cloud Firestore a úložiště Cloud Storage mají pravidla syntaxi popsanou obrázkem 3.5.

[18] Secure Sockets Layer (SSL) – protokol poskytující zabezpečení komunikace
[19] Content delivery network (CDN) – síť vzájemně propojených počítačů
[20] Solid-state Drive (SSD) – elektromagnetický disk
[21] Command Line Interface (CLI) – rozhraní příkazové řádky

```
service <<název služby>> {
    match <<vzor cesty>> {
        allow <<operace>> : if <<podmínka>>;
    }
}
```

Obrázek 3.5: Ukázka syntaxe bezpečnostních pravidel.

Následující ukázka 3.7 definuje pravidla pro databázi Cloud Firestore. Nejdříve zakazuje přístup všem operacím a všem databázovým cestám. Následně tyto pravidla přepisuje pro cestu */courses/courseId*. V rámci této cesty povoluje pravidlo operaci čtení v kolekci courses všem uživatelům. Operaci zápisu povoluje pouze autentizovaným uživatelům a operaci mazání povoluje pouze uživatelům s rolí admin. Ve finále bude přístup ke kolekci courses omezený a přístup ke všem dokumentům ve všech ostatních kolekcích zakázaný.

```
service cloud.firestore {
  match /databases/{database}/documents {

    function isAuthenticated() {
     return request.auth.uid != null;
    }

    function isAdmin() {
     return request.auth.token.ADMIN == true;
    }

    match /{document=**} {
     allow read: if false;
     allow write: if false;
     allow delete: if false;
    }

    match /courses/{courseId} {
     allow read: if true;
     allow write: if isAuthenticated();
     allow delete: if isAdmin();
    }
  }
}
```

Obrázek 3.6: Ukázka syntaxe bezpečnostních pravidel.

3.2.8 Technologie pro komunikace frontend-backend

AngularFire2

AngularFire2 je oficiální knihovna pro Angular podporující Firebase funkcionality. Jedná se o modulární knihovnu podporující různé Firebase funkcionality:

- **AngularFirestoreModule:** Modul pro práci s Firestore databází.
- **AngularFireAuthModule:** Modul pro používání Firebase autentizačních funcionalit.
- **AngularFireStorageModule:** Modul pro práci s datovým úložištěm Firestore.

Dále knihovna obsahuje funkcionality pro spouštění Firebase funkcí. Komunikace mezi klientem a serverem probíhá pomocí protokolu WebChannel. Protokol WebChannel realizuje obousměrnou komunikaci, skrze kterou klient komunikuje se vzdáleným serverem. Protokol umožňuje synchronizaci v reálném čase [2].

3.2.9 Další nástroje a externí služby API

FIO bankovní API

Automatizace rozhraní s Fio bankovním systémem Vám umožní podávání příkazů a získávání dat z účtů vedených u Fio banky. Rozhraní může být použito pro napojení účetních programů nebo pro automatické strojové zpracování pohybů či výpisů z bankovního systému (převzato z [1]).

Možnost napojit se na bankovní API je klíčová pro připravovaný registrační systém neboť umožní automatizovat náročný proces komunikace s klientem ohledně plateb. Veškerá komunikace probíhá pomocí SSL protokolu s minimálně 128bitovým šifrováním [1].

Přístup k datům z účtu vytváří majitel nebo osoba s patřičnými právy ke zvolenému účtu. Ve svém internetovém bankovnictví musí oprávněná osoba vygenerovat token (64 znakový unikátní řetězec). Po vytvoření tokenu lze poté ve frekvenci jednou za pět minut podávat bankovní příkazy nebo stahovat data. Pro podání příkazů nebo stažení dat není nutné být přihlášen do internetového bankovnictví, odpovědi na požadavky se získávají prostřednictvím protokolu HTTPS. API rozhraní má různé metody pro podávání příkazů, získávání strukturovaných dat nebo jejich nastavení [1].

Získání tokenu se provádí následujícími kroky [1]:

1. Oprávněná osoba se přihlásí do internetového bankovnictví FIO banky.
2. Administrace tokenů je přístupná v sekci *Nastavení* v záložce *API*.
3. Požadavek na zřízení tokenu je standardně autorizován. V případě, že je na daném účtu nastavena autorizace vícero osobami, musí token podepsat všechny osoby. Po úspěšném ověření je token zobrazen.
4. Po pěti minutách od úspěšného ověření lze token využívat.

Vlastnosti tokenu [1]:

- Každý token je platný pouze k jednomu bankovnímu účtu.
- Existují dva typy tokenů:
  - **Sledování účtu:** Token je určen pouze k čtení dat z účtu. Odpovídá HTTP metodě GET.
  - **Sledování účtu a zadávání platebních či inkasních příkazů:** Token je určen jak ke čtení dat z účtu, tak i zadávání platebních a inkasních příkazů. Odpovídá HTTP metodě POST.

Získání dat: Komunikace s FIO bankovním systémem probíhá přes protokol HTTPS pomocí metod GET a POST. Získávat data lze v mnoha datových formátech. Pro účely našeho systému bude vhodný formát JSON, neboť je založen na podmnožině programovacího jazyka JavaScript, na kterém je vývoj našeho systému postaven.

```
https://www.fio.cz/ib_api/rest/last/{token}/transactions.json
```

Obrázek 3.7: Ukázka URL pro získání všech pohybů na účtu tokenu *token* od posledního stažení ve formátu JSON.

MailGun[22]

Jedním z požadavků na systém je automatizovaná komunikace s uživateli prostřednictvím emailových zpráv. Za vhodnou službu pro práci s emaily jsem po rešerši možností zvolil MailGun od společnosti RackSpace. Služba nabízí odesílání, přijímání a sledování emailů, tvorbu šablon a další funkce. Základní funkcionality pro naše účely jsou dostupné v potřebném rozsahu zdarma. Všechny funkce jsou dostupné přes REST[23] API nebo použitím tradičních emailových protokolů jako SMTP[24].

[22] https://www.mailgun.com/
[23] Representational State Transfer (REST) – architektura rozhraní pro distribuované prostředí
[24] Simple Mail Transfer Protocol (SMTP) – protokol určený pro přenos emailových zpráv

Kapitola 4

Návrh systému

Při návrhu systému jsem stavěl na analýze případů užití uživatelů systému. Na základě toho byly definovány jednotlivé entity systému. Z entit systému vychází potřebné funkční celky nezbytné pro realizaci celého systému. Ty rozdělujeme na funkce z hlediska uživatele a vnitřní funkce systému. Na závěr shrnuji celou architekturu v kontextu vybraných technologií.

4.1 Návrh entit systému

V kapitole Analýza požadavků jsem pojmenoval několik základních entit stávajícího systému běhu taneční školy. Na základě funkčních požadavků vycházejících z provedené analýzy a analýzy případů užití jsem rozšířil seznam základních entit, které bude potřeba v systému implementovat. Přehled všech entit systému s popisem sepisuji zde:

- **Kurz:** Kurz je pravidelná událost, která se skládá z 1-n lekcí na kterou může student provést registraci.
- **Registrace:** Registraci na daný kurz provádí student, jeden student může mít 0-n registrací.
- **Student:** Student je speciální typ uživatele. Registruje se do 0-n kurzů v daném semestru. V průběhu semestru dochází na lekce kurzů, ve kterých je zaregistrován. Každý student má tedy 0-n docházek na daný kurz.
- **Docházka:** Docházka je záznam o přítomnosti studenta na lekci na dané lekci.
- **Lekce:** Lekce je jedno konání daného kurzu. Každá lekce má 1-2 lektory, kteří ji vedou a 0-n docházek studentů. Každá lekce může mít 0-n video souborů - záznamu z lekce.
- **Lektor:** Lektor je speciální typ uživatele reprezentující učitele vedoucího danou lekci.
- **Pravidelná událost:** Pravidelná údálost je zobecnění kurzu. Jedná se o sadu lekcí, na které není nutná registrace. Každá pravidelná událost se odehrává v rámci jednoho daného semestru pravidelně na jedné dané lokaci. Může být zpoplatněná. Obsahuje mimo jiné popisek a čas konání.
- **Semestr:** Semestr je období v rámci kterého se odehrávají pravidelné události.

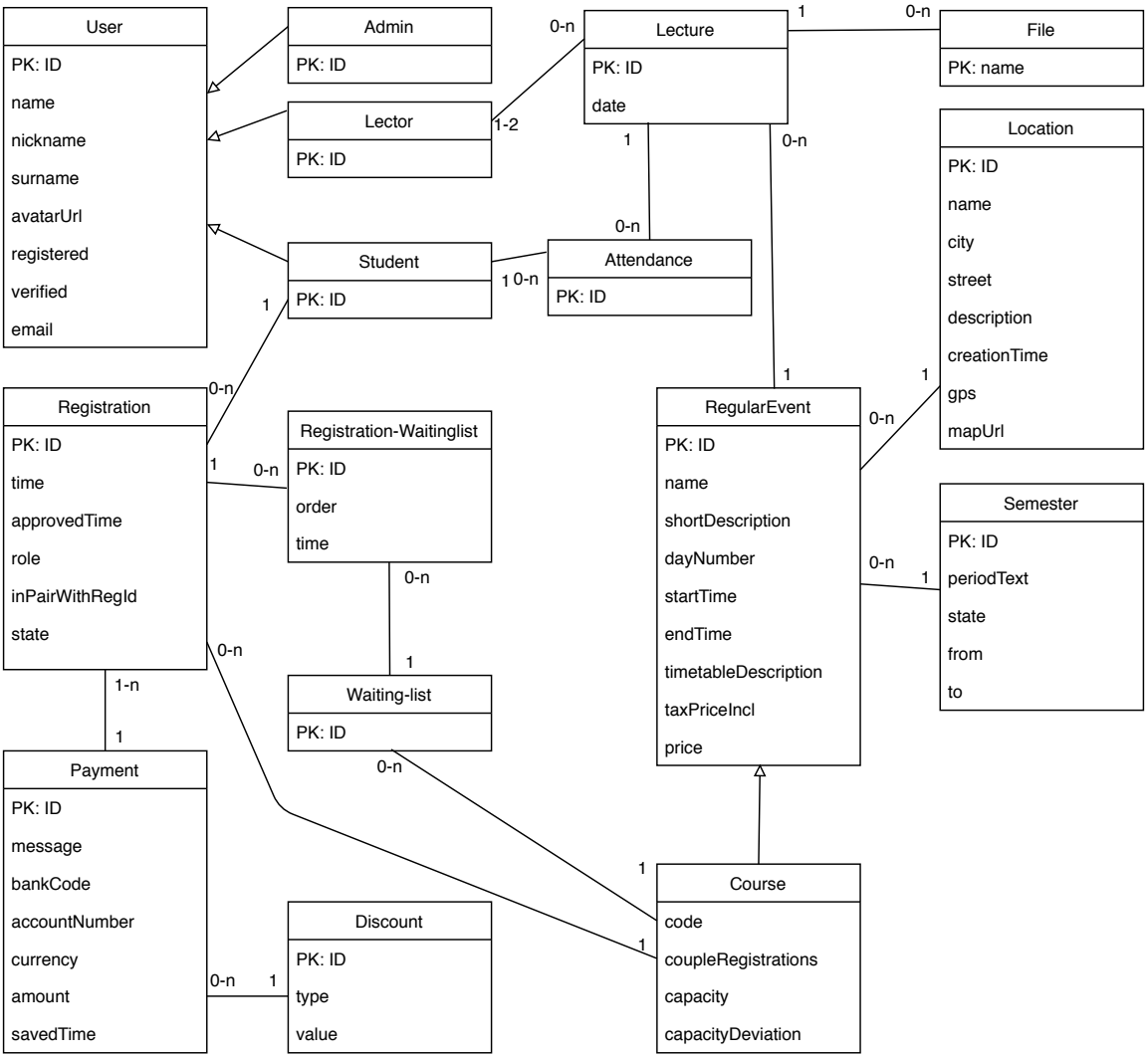
- **Lokace:** Lokace je umístění taneční sálu, kde se v různý čas může odehrávat vícero pravidelných událostí.
- **Uživatel:** Uživatel je zobecnění konkrétních uživatelských rolí systému. Obsahuje jméno, příjmení, přezdívku, obrázek a email uživatele systému.
- **Administrátor:** Administrátor je konkrétní typ Uživatele reprezentující správce systému na straně taneční školy.
- **Čekací listina:** Čekací listina je seznam 0-n registrací na daný kurz, které nebyly do kurzu schváleny z důvodu překročení kapacity kurzu či nevyváženého počtu leaderů a followerů.
- **Platba:** Provedená platba za registrovaný kurz. Může obsahovat údaj o slevě (studentská, párová).

Relace mezi výše popsanými entitami přehledně znázorňuje obrázek entitně relačního diagramu 4.1. Názvy entit jsou uvedeny v angličtině aby odpovídali pojmenování použitému v kódu aplikace. Oproti předchozímu výčtu je zde zaznamenána entita Registration-Waitinglist, která nahrazuje vztah typu m-n mezi entitami Registrace a Kurz.

4.1.1 Denormalizace

Denormalizace je proces přidávání redundantních dat za cílem optimalizování výkonnosti NoSQL databází. Denormalizace umožňuje vyhnutí se komplexnímu spojování dokumentů, zjednodušuje a redukuje počet databázových dotazů. Problém ovšem nastává u konzistence databáze, protože stejná data jsou umístěna na více místech, tudíž pro zachování konzistence se musí změna dat projevit na všech místech kde se redundance vyskytuje. Za tento problém je odpovědný programátor, který musí v tomto případě konzistenci dat zaručit.

V této práci navrhuji denormalizaci aplikovat v kolekci kurzů, kde každý dokument kurzu bude obsahovat redundantní data o lektorech a lokaci.



Obrázek 4.1: Entitně relační diagram.

4.2 Funkční části systému

Na funkční části systému lze pohlížet ze dvou stran. Ze strany uživatele a jeho interakcí s rozhraním systému (frontend) a z hlediska vnitřních funkcí na pozadí systému (backend). V případě frontendu se de facto jedná o čtyři různá rozhraní nabízející funkce pro jednotlivé uživatelské role:

- Přehled kurzů pro studentské registrace v podobě kalendáře
- Nastavení a správa systému pro administrátora
- Podpora práce lektora
- Přehledy pro studenty

Na pozadí systému bude třeba vykonávat následující funkce:

- Synchronizace provedených plateb s FIO bankovním účtem

- Řízení stavu registrací a komunikace se studentem
- Řízení stavu čekacích listin

Jednotlivým funkčním oblastem se věnuji dále podrobněji.

Přehled kurzů v podobě kalendáře

Zájemci se o tanečních kurzech dozvídají skrze webové stránky tanečních škol. Je tedy potřeba propojit stávající web školy s připravovaným systémem. Toto bude provedeno pomocí modulu systému umístitelného do webové stránky taneční školy.

Modul provede zájemce o kurz výběrem jednoho či více vhodných kurzů. Poskytne detailní informace o daném kurzu a umožní provést registraci na kurz zadáním základních údajů (emailu a uplatňované slevy).

Nastavení a správa systému

Manažer taneční školy (zmíněná role administrátor) potřebuje nastavovat parametry systému, sledovat průběh registrací, stavy objednávek a plánovat obsah výukových semestrů.

V rámci plánování obsahu semestrů bude moci administrátor založit v systému daný semestr s parametry datumu začátku a konce období výuky resp. období registrací. Dále má možnost k danému semestru vytvořit sadu nabízených kurzů.

U kurzů lza nastavit jméno, kód a popisek kurzu. Dále cena, zdali se jedná o párový či sólový taneční styl, vyučující lektory, místo a čas konání a datumy konání jednotlivých lekcí pro případ výskytu svátků. Pro snadné používání by mělo jít kurzy mezi semestry kopírovat.

Entita semestr se může nacházet v různých stavech dle toho v jaké fázi přípravy či průběhu semestru se zrovna škola nachází. Identifikoval jsem následujících pět možných fází, které bude dobré v systému zohledňovat:

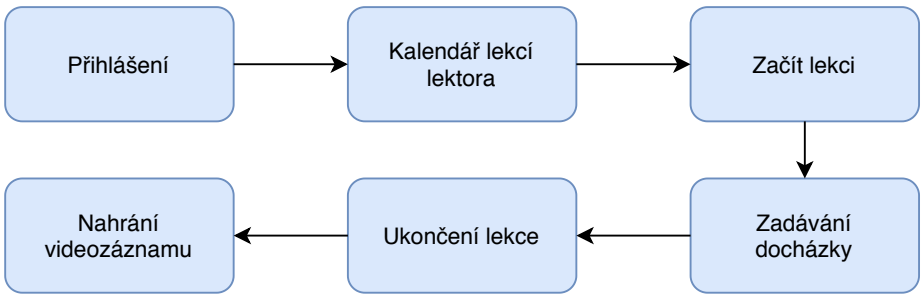
- **Koncept:** Každý nově založený semestr se ocitne ve stavu koncept. Reprezentuje fázi tvorby kurzů a nastavení semestru. V tomto stavu nebude obsah (kurzy) semestru viditelné v modulu kalendáře kurzů.
- **Otevřený:** Ze stavu koncept se kurz dostává do stavu otevřený. Znamená to, že kurzy tohoto semestru jsou přístupné k registracím tedy, že se obsah semestru zobrazuje studentům k možné registraci v rámci modulu kalendáře kurzů.
- **Aktivní:** Jakmile vyprší období registrací do semestru, semestr přejde do stavu aktivní. V aktivním stavu jsou registrace na kurzy již nedostupné a semestr probíhá. V tomto stavu je modul kalendáře kurzů uživatelům dále dostupný, jelikož jej studenti často využívají jako zdroj informací o tom kde a kdy jaký kurz probíhá.
- **Neaktivní:** Semestry v tomto stavu jsou ukončeny, nedá se do nich registrovat, ani se nezobrazují v aplikaci pro registraci. Do stavu neaktivní semestr přejde buďto automaticky po dosažení datumu konce semestru nebo po přepnutí administrátorem.
- **Archivovaný:** Archivované semestry a jejich kurzy budou přesunuty do jiných databázových kolekcí a nebudou se zobrazovat v aktuální nabídce semestrů ke správě.

Obrázek 4.2: Stavy semestrů a přechody mezi nimi.

Podpora práce lektora

Lektor má za úkol kromě samotného vedení lekce i další aktivity. Před lekcí připravuje sál ke konání lekce. Na začátku lekce značí docházku studentů na papír. Na konci lekce pak rekapituluje studentům, co vše se probíralo. Studenti rekapitulaci lekce (obvykle všichni) natáčí. Připravovaný systém v tomto usnadní práci.

Lektor bude mít v rámci aplikace k dispozici rozhraní pro zadávání docházky. Aplikaci bude moci snadno otevřít na svém telefonu či tabletu a nechat studenty se samotné zaznamenat při příchodu do tanečního sálu. Na závěr lekce může celou rekapitulaci natočit jeden ze studentů na lektorovo zařízení a ten pak videozáznam snadno nahraje do systému. Průběh využití systému lektorem znázorňuje obrázek 4.3.



Obrázek 4.3: Průběh využití systému lektorem.

Přehledy pro studenty

Student bude mít v rámci systému dostupné rozhraní, skrze které bude dostupný výpis stavu objednávek (registrací na kurzy) a provedených plateb. Student zároveň může ve svém rozhraní zobrazit videa z absolvovaných lekcí. Dále si bude moci zobrazit kalendář, kde se mu zobrazí jeho události (lekce kurzů).

Řízení stavu registrací a komunikace se studentem

Hlavní funkcí systému je řízení procesu registrací studentů. Na registraci studenta se můžeme dívat tak, že se pohybuje mezi různými stavy. Po vyplnění registračního formuláře studentem v rámci modulu kalendáře kurzů dojde k odeslání dat z frontendu systému na backend. Jako první krok registrace je třeba ověřit zdali uživatelem zadaný email existuje.

Na základě (ne)existence vzniká buďto ověřená či neověřená registrace. Neověřená registrace čeká na ověření emailové adresy studenta. Jakmile k tomu dojde stává se z ní ověřená registrace a přechází k dalšímu zpracování, ke kterému dojde v případě, že je student ověřen.

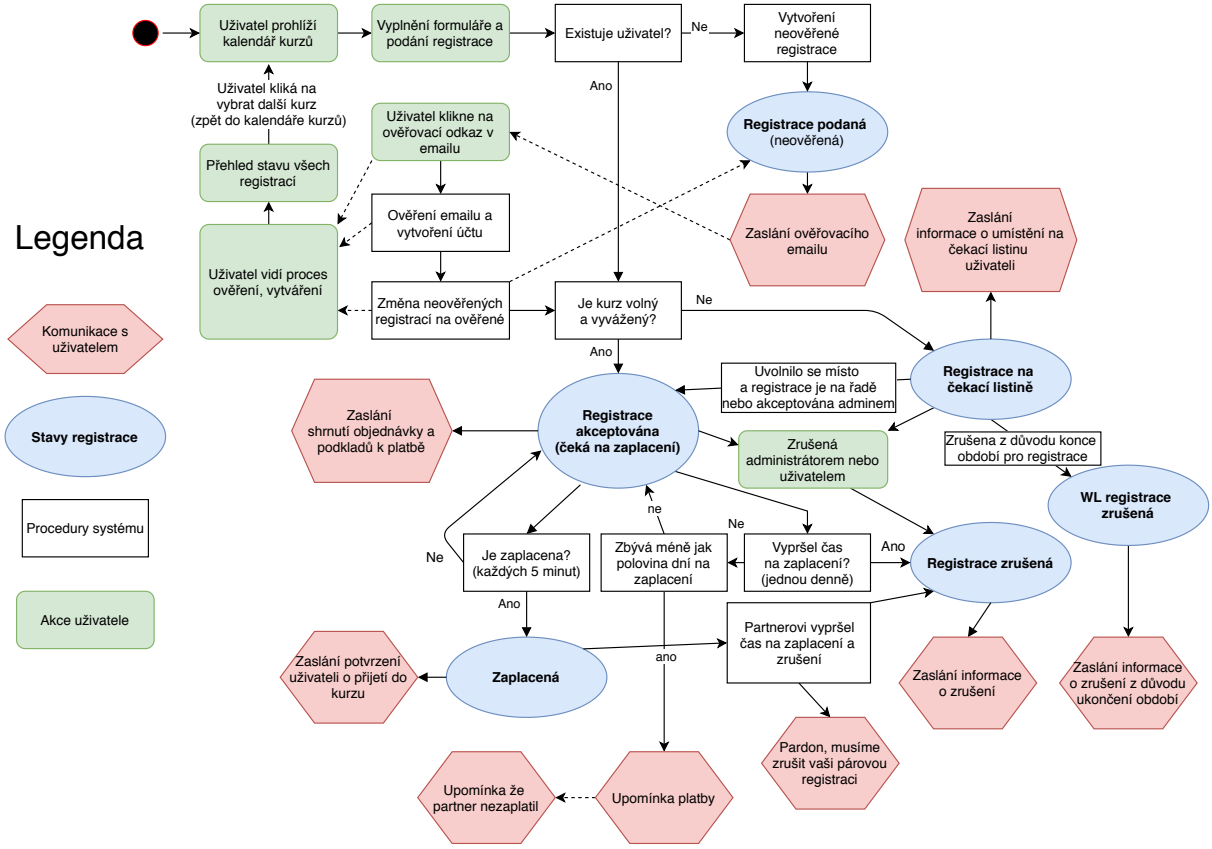
Systém v tomto kroku bude ověřovat naplněnost a vyváženost daného kurzu. V případě, že je v kurzu místo a je vyvážený, dojde automaticky k akceptaci studenta do kurzu. Pokud

je ale kurz naplněn či nevyvážen (z hlediska leaderů a followerů), dojde k umístění registrace na čekací listinu.

Na čekací listině mohou nastat dvě situace. Může dojít k uvolnění kurzu a tedy k posunu registrace blíže akceptaci či rovnou akceptaci nebo dojde ke zrušení registrace. Ke zrušení registrace na čekací listině může dojít rozhodnutím studenta, administrátora nebo v situaci, kdy doběhne období registrací do daného semestru.

Ve chvíli, kdy je registrace akceptována, dojde k vygenerování platebních údajů a zaslání na email studenta. Tímto okamžikem začne běžet nastavitelný časový limit pro uhrazení registrace převodem na účet. Systém pravidelně kontroluje stav platby na propojeném bankovním účtu. Pokud zaregistruje příchozí platbu odpovídající vygenerovaným platebním údajům, posune registraci do stavu zaplacená. Kromě toho systém také jednou denně bude kontrolovat, zda nevypršel čas na zaplacení. Pokud ano, dojde k automatickému zrušení registrace. Pokud časový limit pro platbu nevypršel, systém zašle uživateli upozornění.

Celý proces registrace je přehledně popsán na obrázku 4.4. Při jakékoliv změně stavu registrace dochází k notifikování uživatele na jeho email.



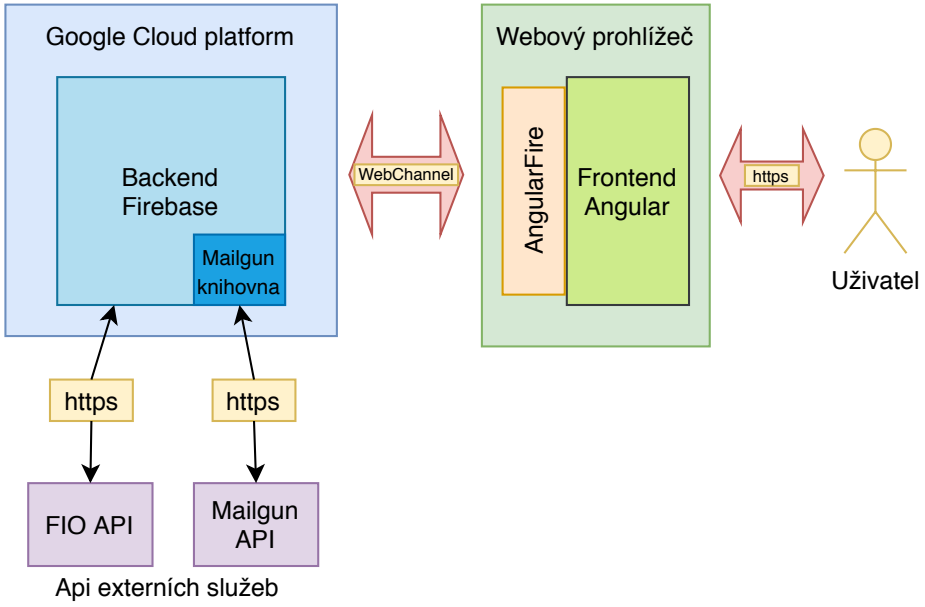
Obrázek 4.4: Proces registrace studenta na kurz.

Synchronizace provedených plateb s FIO bankovním účtem

Aby mohl systém plně automaticky řídit stavy registrací, je nutné aby byl propojený s bankovním účtem spolupracující taneční školy. Bude nutné pravidelně v krátkých intervalech stahovat z bankovního účtu data o příchozích platbách a párovat je s registracemi.

4.3 Architektura systému

Obrázek 4.5 zobrazuje architekturu celého systému. Ve webovém prohlížeci se bude uživateli zobrazovat prezentační část a to buď modul pro registraci do kurzů, nebo administrátorské rozhraní. Obě frontendové aplikace budou pomocí knihovny AngularFire a protokolu WebChannel komunikovat s Firebase backendem. Backend bude komunikovat pomocí HTTPS s externími službami.



Obrázek 4.5: Architektura systému.

4.4 Grafické uživatelské rozhraní

Součástí návrhu systému jsou obrazovky, které bude systém zobrazovat na základě vstupních dat. Při návrhu grafického uživatelského rozhraní je kladen důraz na vytvoření intuitivního a jednoduchého ovládání systému. Následuje výpis obrazovek pro rozhraní registrace:

- **Obrazovka pro rozvrh kurzů a semestrů:** Obrazovka obsahuje záložky představující aktivní semestry a semestry otevřené k registracím. Každá tato záložka zobrazuje týdenní rozvrh kurzů v daném semestru, který přehledně informuje uživatele o dni, místě a času konání, lektorech kurzu a o dalších volitelných informacích.
- **Obrazovka detailu kurzu:** Obrazovka se zobrazí po kliknutí na vybraný kurz. Obrazovka zobrazuje detailnější informace o kurzu jako název, popis, počet lekcí, základní cenu, kapacitu a aktuální vyvážení rolí.
- **Obrazovka registračního formuláře:** Tuto obrazovku aktivuje kliknutí na tlačítko Mám zájem o kurz dostupné v obrazovce detailu kurzu. Obrazovka umožňuje vyplnit a odeslat registrační formulář s možností výběru slev. Po aplikaci slevy se automaticky přepočítává cena.

- **Obrazovka stavu registrace:** Cílem této obrazovky je informovat uživatele o jeho odeslané registraci a případně o dalších krocích k potvrzení, nebo zaplacení objednávky.

Všechny výše zmíněné obrazovky, kromě obrazovky pro rozvrh kurzů a semestrů, budou zobrazovány v postranním panelu, který se po kliknutí na vybraný kurz zobrazí na pravé straně obrazovky.

Následuje výpis obrazovek pro administrátorské rozhraní:

- **Obrazovka pro přihlášení:** Obrazovka se zobrazí vždy před vstupem do administračního systému. Nabídne dvě vstupní pole pro přihlašovací email a heslo.
- **Obrazovka pro obnovení hesla:** Obrazovka umožňuje uživateli obnovit zapomenuté heslo ke svému účtu.
- **Obrazovka pro registraci do systému:** Obrazovka umožňuje uživatelům registrovat se do systému.
- **Obrazovka pro ověření emailu a automatické vytvoření studentského účtu:** Tato obrazovka je dostupná po kliknutí na odkaz v ověřovacím emailu, který uživateli přišel po uskutečnění první objednávky. Po automatickém ověření se uživateli vytvoří účet, do kterého je automaticky přihlášen s vygenerovaným heslem. Toto heslo se uživateli okamžitě odešle na email.
- **Obrazovka pro doplnění povinných údajů:** Tato obrazovka je vidět po přihlášení uživatele, který nemá vyplněné povinné údaje. Z této obrazovky se nedá dostat jinak, než vyplněním a odesláním těchto údajů.
- **Obrazovka pro správu kurzů:** Obrazovka umožňuje veškerou správu všech semestrů a všech kurzů v nich.
- **Obrazovka pro správu objednávek:** Obrazovka umožňuje zobrazit všechny objednávky v systému a měnit jejich stavy.
- **Obrazovka pro přijaté platby:** Tato obrazovka zobrazuje jednotlivé platby, které byly systémem přijaty.
- **Obrazovka pro správu lektorů:** Obrazovka zobrazuje všechny uživatele s rolí lektor. Dále umožňuje pozvat nové lektory do systému, odebrat lektorské role a odstranit lektora z vybraných, jeho aktuálně vyučovaných, kurzů.
- **Obrazovka pro správu lokací:** Obrazovka poskytuje přehled o všech lokacích. Dále umožňuje vytvářet nová, a upravovat a mazat stávající.
- **Obrazovka pro správu uživatelů:** Cílem obrazovky pro správu uživatelů je vyobrazení veškerých registrovaných uživatelů v systému. Uživatelům lze zde přidávat jednotlivé role.
- **Obrazovka pro zobrazení kalendáře:** Tuto obrazovku sdílí uživatelé s rolí student a lektor. Kalendář pro lektora zobrazuje lekce kurzů, ve kterých je lektorem. Pro studenta se zobrazují lekce kurzů, do kterých se zaregistroval a ve kterých tuto objednávku zaplatil. V případě, že uživatel má obě zmíněné role, jsou události, ve kterých je lektorem a ve kterých je studentem, barevně rozlišené.

- **Obrazovka pro docházku:** Docházka se bude typicky zadávat na mobilním zařízení, proto lektoři pro lepší práci s docházkou mohou tuto obrazovku přepnout do režimu celé obrazovky. Lektoři mohou prostřednictvím této obrazovky k lekci nahrát výukové videa.
- **Obrazovka pro videa z lekcí:** Obrazovka poskytuje přehled o dostupných výukových videí z lekcí kurzů, kterých se student účastní. Videa si může jednoduše procházet a sledovat.
- **Obrazovka pro přehled objednávek uživatele:** Tato obrazovka zobrazuje přehled objednávek, které přihlášený uživatel vytvořil.
- **Obrazovka pro přehled plateb uživatele:** Cílem obrazovky je poskytnout přihlášenému uživateli přehled o jeho uskutečněných platbách.
- **Obrazovka pro správu uživatelského profilu:** Tato obrazovka dává možnost přihlášenému uživateli změnit si své jméno, příjmení, přezdívku, heslo a profilový obrázek.

Součástí každé obrazovky v administrátorském rozhraní (kromě přihlašovací, registrační, ověřovací, pro obnovu hesla a pro docházku) je postranní navigační panel, který umožňuje snadnou orientaci v administrační aplikaci. Panel obsahuje odkazy na všechny ostatní obrazovky. Pro přihlášeného uživatele se zobrazují jen ty odkazy, které odkazují na obrazovky, do kterých má přihlášený uživatel práva vstoupit. Pokud je přihlášeným uživatelem administrátor, zobrazí se mu odkazy na obrazovky pro správu kurzů, objednávek, přijatých plateb, lektorů, lokací a uživatelů. Pokud je přihlášen uživatel s rolí lektora, je mu zobrazen odkaz na obrazovku kalendáře. Pokud je role přihlášeného uživatele student, zobrazí se odkazy na obrazovky pro kalendář, videa z lekcí, objednávky a platby. V případě že má uživatel více rolí, jsou mu zobrazeny všechny příslušné odkazy. Všichni přihlášení uživatelé mohou přes postranní panel zobrazit obrazovku svého profilu. Dále se mohou v postranním panelu odhlásit.

Všechny zmíněné obrazovky (kromě té profilové) zobrazují formuláře pomocí modálních oken. Pokud tedy dochází k vytváření záznamu, jeho editaci nebo zobrazení detailu, jsou tyto formuláře zobrazeny v modálních oknech. Okno je odstíněno od původní obrazovky, které je vidět v pozadí.

Kapitola 5

Implementace

Podle návrhu uvedeném výše byl vytvořen registrační systém pro swingovou taneční školu. Frontendová část je napsána ve značkovacím jazyce HTML5, nastylována pomocí CSS3 a aplikační logika systému je vytvořena v programovacím jazyce TypeScript s využitím frameworku Angular ve verzi 8.3.19. Serverová část je napsána v programovacím jazyce JavaScript v prostředí Node.js ve verzi 10, které je nasazeno v cloudové službě Firebase.

5.1 Frontend

V této podkapitole popisuji propojení Angular projektu s Firebase projektem (propojení frontend - backend). Dále strukturu, nasazení a zabezpečení aplikace modulu pro registrace a aplikace administrátorského rozhraní.

5.1.1 Připojení k backendu

Aby aplikace mohla komunikovat s backendem, je potřeba nakonfigurovat Angular projekt. Komunikaci zajišťují knihovny AngularFire2 a Firebase, které je třeba nejdříve nainstalovat do Node.js prostředí pomocí npm[1] příkazu na obrázku 5.1.

```
npm install angularfire2 firebase --save
```

Obrázek 5.1: Příkaz pro nainstalování AngularFire2 a Firebase knihoven v prostředí Node.js.

Následně je nutné vložit Firebase konfiguraci 5.2 do vývojového a produkčního prostředí Angular projektu. Tato konfigurace obsahuje unikátní identifikátory, může se tedy zdát, že vložení této konfigurace do projektu představuje velké bezpečnostní riziko, protože po nasazení bude tato konfigurace veřejná. Tato akce by byla velkým bezpečnostní rizikem, pokud by se backend nezabezpečil pomocí security rules. Jelikož je backend zabezpečený (popsáno níže), tak tato akce bezpečnostní riziko nepředstavuje.

Finálním krokem je importování AngularFire, Firebase modulů a proměnných prostředí do kořenového modulu aplikace *AppModule* 5.3.

Tímto je aplikace připravená na implementaci funkcionalit vyžadující spojení s Firebase službami.

[1] Node Package Manager (npm) – správce balíčků pro prostředí Node.js

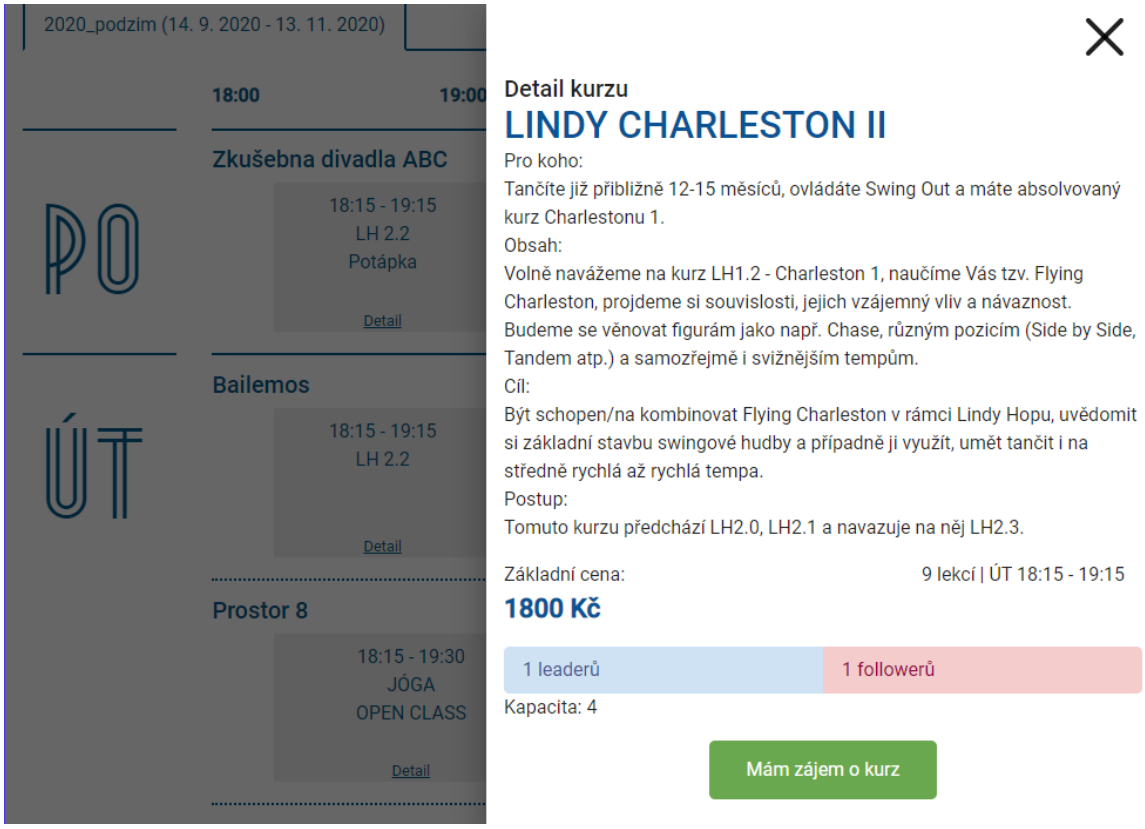
```
firebaseConfig: {
    apiKey: 'AIzaSyA_hscs6PDDwRKqBi4k80LsEjtN0V6CZ84',
    authDomain: 'swing-school.firebaseapp.com',
    databaseURL: 'https://swing-school.firebaseio.com',
    projectId: 'swing-school',
    storageBucket: 'swing-school.appspot.com',
    messagingSenderId: '902778048133',
    appId: '1:902778048133:web:1264a3aa7f0ed7954ee3ec',
    measurementId: 'G-4RNQKK1CHQ'
}
```

Obrázek 5.2: Firebase konfigurace.

5.1.2 Modul pro registrace

Jelikož směrování používá URL, mohlo by se stát, že aplikace a stávající systém školy, na které bude aplikace doplňkem, budou pracovat se stejným URL, což by vedlo k problémům. Při implementaci aplikace pro rozhraní registrace jsem se tedy použití směrování (routing) vyhnul.

Snímek z aplikace rozhraní registrace je na obrázku 5.4.



Obrázek 5.4: Snímek obrazovky z aplikace pro rozhraní registrace.

```
import { AngularFireModule } from '@angular/fire';
import { AngularFirestoreModule } from '@angular/fire/firestore';
import { AngularFireStorageModule } from '@angular/fire/storage';
import { AngularFireFunctions } from '@angular/fire/functions';
import { AngularFireAuthModule } from '@angular/fire/auth';
import { environment } from 'src/environments/environment';
import * as firebase from 'firebase/app';

@NgModule({
 declarations: [
   AppComponent
 ],
 imports: [
   BrowserModule,
   AngularFireModule.initializeApp(environment.firebaseConfig),
   AngularFirestoreModule,
   AngularFireStorageModule,
   AngularFireAuthModule,
 ],
 providers: [
   AngularFireFunctions,
 ],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

Obrázek 5.3: Import Firebase a AngularFire modulů do kořenového modulu aplikace.

Struktura

Rozsah aplikace rozhraní pro registrace není velký, není tedy třeba aplikaci členit do více modulů. V tomto případě stačí jeden hlavní modul. V následující struktuře popisuji uspořádání všech vytvořených komponent a modulu v aplikaci:

- **app.component** je nejvýše postavená komponenta, zobrazuje se jako první. Připravuje postranní panel ve kterém zobrazí komponenty provádějící registrací. Dále zobrazuje komponentu "callendar".
  - **back-button.component** obsahuje pouze tlačítko, které se zobrazuje při registračním formuláři. Kliknutím se uživatel dostane zpět na zobrazení detailu kurzu.
  - **callendar.component** zobrazuje přehledný týdenní rozvrh vypsaných kurzů.
  - **callendar-course.component** zobrazuje dílčí kurz v rozvrhu.
  - **course-detail.component** je detail kurzu v postranním panelu, který se zobrazí po kliknutí na kurz v rozvrhu.
  - **course-heading.component** zobrazuje především název kurzu a stav registrace.

- **course-price.component** se stará o správné zobrazení ceny objednávky. Na základě výběru slev zjišťuje z backendu finální cenu.
- **course-registration.component** obsahuje registrační formulář.
- **course-registration-status.component** informuje uživatele o stavu jeho podané registrace a případně o následujících krocích pro akceptaci registrace.
- **cross-button.component** je tlačítko pro schování postranního panelu.
- **register-button.component** je tlačítko, které zobrazí registrační formulář.
- **role-balance.component** je vizuální prvek informující o počtech registrovaných leaderů a followerů v kurzu.
- **selected-course-sub-info.component** je komponenta zobrazující den a čas konání vybraného kurzu, počet lekcí v kurzu a datum začátku a konce semestru.
- **sidebar-wrap.component** obaluje komponenty provádějící uživatele registrací (course detail, course-registration, course-registration-status).

Nasazení

Stávající systém školy funguje na redakčním systému WordPress. Způsoby nasazení aplikace do systému školy se nabízejí dva:

- **Lokálně:** Pomocí nahrání zdrojových souborů registračního systému do stávajících webových stránek spolupracující taneční školy a importování do kódu souboru stávajícího webu, který se stará o zavedení celé aplikace.
- **Vzdáleně:** Nahrát aplikaci na Firebase hosting a vložit do školního systému HTML element *iframe* s atributem *src* odkazující na doménu hostingu. Element iframe umožňuje ve webové stránce zobrazit jinou webovou stránku. Tyto dokumenty mezi sebou nesdílí kaskádové styly, proměnné, ani jiné prvky.

Obě varianty jsem vyzkoušel a zjistil následující výhody a nevýhody:

- **Lokálně:**
  - Výhody:
    * Správce školního webu může měnit styly modulu.
  - Nevýhody:
    * Při každé aktualizaci aplikace je nutné do školního systému nahrávat nové zdrojové soubory.
    * Styly školního systému přepisují styly aplikace.
- **Vzdáleně:**
  - Výhody:
    * Při aktualizaci aplikace se změny aplikují velice snadno.
    * Správce školního systému může iframe element jednoduše stylovat a přemisťovat v rámci jejich webu.
  - Nevýhody:

* Jelikož obsah elementu iframe je jiný dokument, není možné aby rodičovský dokument (školní systém) bez další pomoci odhadl výšku obsahu elementu. Zobrazení aplikace by tedy bylo vždy buď s posuvníkem (scrollbar), nebo se zbytečným odsazením. Navíc se tato výška může měnit, protože aplikace je responzivní, tudíž při změně šířky okna se může měnit výška obsahu.

Nevýhoda vzdáleného řešení se ovšem dá vyřešit vytvořením dvou níže uvedených skriptů (5.5, 5.6), které zajistí komunikaci mezi nadřazeným dokumentem (aktuální školní web) a podřazeným dokumentem (modul rozhraní registrace). Při této komunikaci posílá podřazený dokument dokumentu nadřazenému číslo, které odpovídá výšce jeho obsahu v pixelech. Nadřazený dokument tuto výšku nastaví elementu iframe a tím se zajistí to, že výška obsahu je vždy stejná s výškou elementu.

```
<script>
    window.parent.postMessage(this.scheduleHeight, '*');
</script>
```

Obrázek 5.5: Skript pro odeslání výšky obsahu do nadřazeného dokumentu.

```
<script>
   function bindEvent(element, eventName, eventHandler) {
      if (element.addEventListener){
         element.addEventListener(eventName, eventHandler, false);
      } else if (element.attachEvent) {
         element.attachEvent('on' + eventName, eventHandler);
      }
   }
   bindEvent(window, 'message', function(e) {
      document.getElementById('iframe').style.height = e.data + 'px';
   });
</script>
```

Obrázek 5.6: Skript pro získání výšky obsahu podřazeného dokumentu a nastavení výšky elementu iframe.

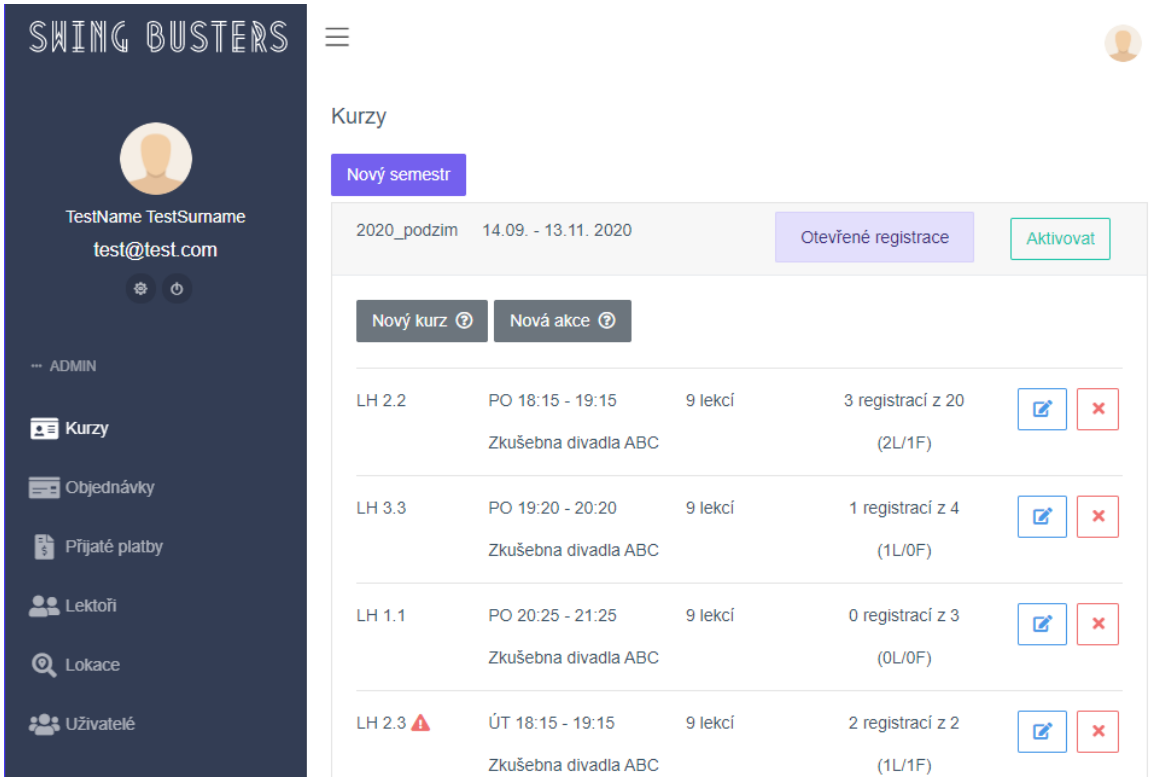
Na základě výše popsaných výhod a nevýhod jsem zvolil nasazení vzdáleným způsobem.

Zabezpečení

Modul pro registrace zobrazuje pouze informace o kurzech v aktuálních semestrech a umožňuje registraci na ně. Celý modul je veřejně přístupný komukoli. Zabezpečení registrací je vyřešeno v backendové části. Není tedy potřeba tuto frontendovou část nijak zvlášť zabezpečovat.

5.1.3 Administrátorské rozhraní

Snímek z aplikace administrátorského rozhraní je na obrázku 5.7.



Obrázek 5.7: Snímek obrazovky z aplikace pro rozhraní registrace.

Struktura

Oproti rozhraní pro registraci je administrátorské rozhraní rozsahem větší a komplexnější. Je tedy třeba členit tuto aplikaci do více modulů a také je potřeba použít směrování. Následuje výpis jednotlivých modulů a popis jejich komponent:

- **app.module** je kořenový modul celé aplikace.
  - **app.component** je nejvýše postavená komponenta. Na základě směrovače zobrazuje příslušné moduly.
- **attendance.module** je docházkový modul.
  - **attendance.component** je komponenta pro zadávání docházky lekce.
  - **upload-video-modal.component** slouží k nahrávání výukových videí k lekci.
- **authentication.module** je autentizační modul.
  - **404.component** se zobrazí při zadání takového URL, které neodpovídá žádné cestě v aplikaci.
  - **complete-registration.component** slouží pro dokončení registrace. Typicky doplnění jména, příjmení a přezdívky při registraci uživatele.
  - **lector-invitation.component** se zobrazí při kliknutí na odkaz v emailu s pozvánkou.
  - **login.component** je komponenta pro přihlášení do systému.

  - **signup.component** je komponenta pro registraci do systému.

- **calendar.component** zobrazuje kalendář událostí.
  - **calendar-utils.component** slouží pro úpravu kalendáře.
- **course-detail.component** je modální okno s detaily události. Typicky se zobrazuje po kliknutí na událost v kalendáři.
- **courses.module** je modul pro správu semestrů a kurzů.
  - **course-dialog.component** zobrazuje modální okno pro vytváření nebo editaci kurzu nebo akce.
  - **course-list.component** obaluje jednotlivé komponenty kurzů.
  - **course-list-item.component** zobrazení pro jednotlivé kurzy.
  - **courses-wrap.component** obaluje všechny kurzy ve všech semestrech.
  - **lectures-dialog.component** zobrazuje modální okno pro správu lekcí v daném kurzu.
  - **new-semester-dialog.component** je modální okno pro vytvoření nového semestru. Při vytváření nového semestru je možné vybrat semestr ze kterého se duplikují všechny kurzy do nového.
  - **semester.component** obaluje všechny kurzy z jednoho semestru a přidává tlačítka na změnu stavu semestru a vytvoření nových kurzů či akcí.
- **info.module** modul pro další informace a ověřování.
  - **verification.component** se zobrazí po kliknutí na tlačítko v emailu pro potvrzení. Komponenta se stará o potvrzení emailu a automatické vytvoření nového účtu.
- **lectors.module** je modul lektorů.
  - **lectors-dialog.component** modální okno, které umožňuje přidání lektorské role uživateli, spravování vyučování lektora a pozvání nového lektora.
  - **lectors-list.component** obaluje jednotlivé komponenty lektorů.
  - **lectors-list-item.component** zobrazení pro jednotlivé lektory v seznamu.
  - **lectors.component** obaluje seznam lektorů a přidává tlačítko pro zobrazení okna pro správu lektorů.
- **lecture-videos.module** je modul pro zobrazení a přehrání výukových videí z lekcí.
  - **video-modal.component** modální okno zobrazující vybrané video.
  - **lecture-videos.component** zobrazuje seznam všech dostupných videí k dané lekci.
- **locations.module** je modul pro správu lokací.
  - **location-dialog.component** je modální okno pro vytvoření nové nebo editování stávající vybrané lokace.
  - **location-list.component** obaluje jednotlivé komponenty lokací.

  - **location-list-item.component** zobrazení pro jednotlivé lokace v seznamu.
  - **locations.component** obsahuje seznam lektorů a přidává tlačítko pro zobrazení okna pro správu lokací.

- **my-orders.module** je modul pro zobrazení objednávek studenta.
  - **my-orders.component** zobrazuje seznam objednávek přihlášeného studenta.
- **my-payments.module** je modul pro zobrazení studentských plateb přijatých v systému
  - **my-payments.component** zobrazuje seznam plateb přihlášeného studenta.
- **orders.module** je modul pro správu všech objednávek v systému.
  - **orders.component** zobrazuje seznam všech objednávek, které systém přijal.
- **payments.module** je modul pro správu všech plateb přijatých systémem.
  - **payments.component** zobrazuje seznam všech plateb, které systém přijal.
- **profile.module** je modul pro správu uživatelského profilu.
  - **profile.component** je komponenta ve které může přihlášený uživatel změnit jméno, příjmení, přezdívku, heslo, profilový obrázek a v případě neověřeného emailu odeslat nový ověřovací email.
- **schedule.module** je modul pro kalendář studentů a lektorů.
  - **lector-modal.component** je modální okno, které se zobrazí když lektor klikne v kalendáři na událost. Okno dává lektorovi na výběr ze dvou akcí:
    * Otevřít docházku
    * Zobrazit detail kurzu
  - **schedule.component** obaluje komponentu kalendáře.
- **shared.module** je modul obsahující sdílené komponenty.
  - **spinner.component** zobrazuje animaci načítání.
  - **sidebar.component** zobrazuje postranní panel.

Zabezpečení

Stránku administrátorského rozhraní mohou otevřít celkově dva typy uživatelů:

- Neregistrovaný uživatel bez role
- Registrovaný uživatel s různými rolemi

Jelikož toto rozhraní využívá směrování, je nutné zajistit to, aby každý výše zmíněný typ uživatele mohl navštívit pouze tu část rozhraní, do které má právo přistoupit. O toto se starají takzvané *router guards*. Jsou to funkce, které směrovači říkají zda má nebo nemá umožnit požadovanou změnu cesty.

Tyto ochranné funkce jsem implementoval dvě:

- authGuard: Funkce přesměruje nepřihlášeného uživatele přesměruje na obrazovku přihlášení.
- roleGuard: Funkce nedovolí požadovanou změnu cesty pokud uživatel nemá dostatečné oprávnění.

Nasazení

Nasazení aplikace administrátorského rozhraní proběhlo pomocí Firebase hostingu na přidělených subdoménách na doménách *web.app* a *firebaseapp.com*.

5.1.4 Použité externí komponenty a knihovny

Při tvorbě aplikace byly použity tyto knihovny a sady nástrojů:

- **AdminBite Angular Template**[2] Angular šablona administrátorského rozhraní. Z této šablony byly využity především obrazovky pro přihlášení a registraci, komponenty postranního menu, tlačítek a formulářů.
- **Angular Material**[3]**:** Angular Material je knihovna komponent uživatelského rozhraní pro Angular. Tyto komponenty jsou tvořené podle specifikací materiálového designu od Google.
- **Bootstrap**[4]**:** Bootstrap je CSS framework, který obsahuje šablony pro různé komponenty rozhraní. Z rozsáhlé sady nástrojů, které Bootstrap poskytuje byl v tomto projektu nejpodstatnější takzvaný mřížkový systém. Ten funguje společně se zlomovými body. V tomto mřížkovém systému lze jednotlivým elementům definovat chování, při kterých se bude měnit struktura elementů na základě šířky okna.
- **Font Awesome**[5]**:** Sada fontů, které mají ikony místo znaků. Díky vektorové reprezentaci lze ikony jednoduše upravovat pomocí CSS bez ztráty kvality. Tyto ikony se používají například v menu aplikace.
- **ngx-toastr**[6]**:** Knihovna pro neblokující notifikace. Tato komponenta je využita pro zobrazení stavu všech operací, které komunikují s backendem.
- **Angular Calendar**[7]**:** Komponenta pro Angular umožňující zobrazit události v kalendáři. Tato komponenta studentům a lektorům zobrazuje lekce registrovaných kurzů studenta, případně lekce kurzů, které lektor vyučuje.

5.2 Backend

V této kapitole popisuji zpracování registrací, čekací listinu, zajištění konzistence dat v databázi pomocí transakčního zpracování, strukturu všech backendových funkcí a zabezpečení backendu.

[2] https://www.wrappixel.com/demos/angular-admin-templates/adminbite-angular/docs/documentation.html:
[3] https://material.angular.io/
[4] https://getbootstrap.com/
[5] https://fontawesome.com/
[6] https://www.npmjs.com/package/ngx-toastr
[7] https://github.com/mattlewis92/angular-calendar

5.2.1 Zpracování registrací

Registrace jsou v systému rozlišovány na tři druhy:

- **Registrace jednotlivce na kurz pro jednotlivce**
- **Registrace jednotlivce na párový kurz**
- **Registrace páru na párový kurz**

V těchto druzích mohou nastat následující situace:

Registrace jednotlivce na kurz pro jednotlivce:

- **Uživatel se na kurz už zaregistroval:** Registrace není vytvořena.
- **Kurz je volný:** Registrace je akceptována a zařazena do kurzu.
- **Kurz je plný:** Registrace je přijata, ale přesouvá se na poslední místo v čekací listině.

Registrace jednotlivce na párový kurz:

- **Uživatel se už na kurz registroval:** Registrace není vytvořena.
- **Kurz je volný a pro danou roli vyvážený (rozdíl v počtu rolí je menší nebo roven jak maximální odchylka):** Registrace je akceptována a zařazena do kurzu.
- **Kurz je volný, ale pro danou roli nevyvážený:** Registrace je přijata, ale přesunuta na poslední místo v čekací listině dané role.
- **Kurz je plný:** Registrace je přijata, ale umístěna na poslední místo v čekací listině dané role.

Registrace páru na párový kurz:

- **Alespoň jeden z uživatelů se už na kurz registroval:** Registrace nejsou vytvořeny.
- **Kurz je volný a vyvážený pro alespoň jednu roli z páru:** Registrace jsou akceptovány a zařazeny do kurzu a to i v případě, že v kurzu je pouze jedno volné místo. V tomto případě je povoleno překročit limit kapacity kurzu.
- **Kurz je plný:** Registrace jsou přijaty, ale umístěny na poslední místa v čekacích listinách daných rolí.

5.2.2 Čekací listina

Čekací listina obsahuje záznamy o objednávkách, pro které při registraci nebyly místa v daných kurzech. Pro správné určení pozice objednávky ve frontě čekací listiny se ukládá čas vytvoření objednávky a pořadí dané role v rámci čekací listiny kurzu. Díky těmto informacím se později dá určit která objednávka se bude z čekací listiny posouvat do kurzu.

Kontola posuvu objednávek z čekací listiny do kurzu se provádí vždy při těchto operacích:

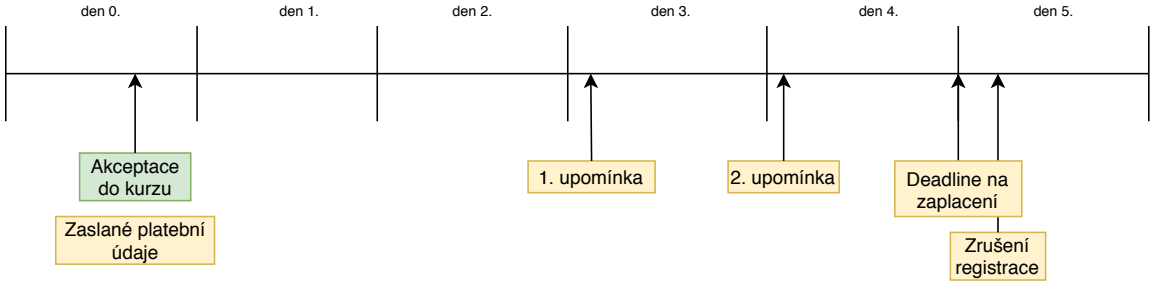
- Vytvoření nové objednávky v kurzu
- Zrušení objednávky v kurzu
- Změna kapacity kurzu
- Změna maximální odchylky rolí v kurzu

V případě posunu registrace, která je párová (je propojená s další registrací), se automaticky posouvá i ta druhá, párová registrace. V tomto případě je povoleno přesáhnout kapacitu kurzu o jedno místo.

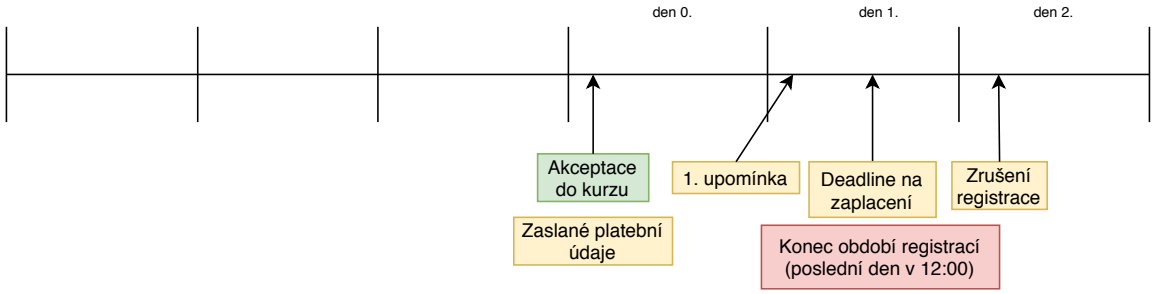
5.2.3 Termín a upomínky zaplacení

V případě, že je registrace akceptována (je zařazena do kurzu) více jak čtyři dny před koncem období registrací, jsou studentovi zaslány platební údaje. Student má na zaplacení objednávky čtyři pracovní dny. Od poloviny tohoto časového intervalu systém studentovi automaticky zasílá každé ráno upomínkové emaily na zaplacení objednávky. Pokud student objednávku v tomto intervalu nezaplatí, je mu následující den ráno objednávka automaticky zrušena. Tyto situace popisuje obrázek 5.8.

Podobné chování je i v případě, že je registrace akceptována méně jak čtyři dny před koncem období registrací. Rozdíl je v tom, že termín zaplacení nejsou čtyři pracovní dny, ale termín konce období registrací. Tuto situaci popisuje obrázek 5.9.



Obrázek 5.8: Proces včasné registrace (déle jak čtyři pracovní dny před koncem období registrací).



Obrázek 5.9: Proces registrace před koncem období registrací (méně jak čtyři pracovní dny před koncem období registrací).

5.2.4 Transakční zpracování

Aby byla zajištěna konzistence dat v databázi, je nutné využívat při manipulaci s daty transakční zpracování. Narozdíl od tradičního SQL[8], kde je zahájení transakcí implicitní [16], ve Firebase databázi musí transakce vytvářet programátor. Každá transakce ve Firebase se skládá z posloupností operací čtení a zápisů. Přičemž je nutné zachovat toto pořadí: nejprve všechny operace čtení a poté všechny operace zápisů (vizte obrázek 5.10).

```
let transaction = db.runTransaction(t => {
  const registrationOneRef = db.doc(‘registrations/${regOneId}‘);
  const registrationTwoRef = db.doc(‘registrations/${regTwoId}‘);
  await t.get(registrationOneRef);
  await t.get(registrationTwoRef);
  t.update(registrationOneRef, { state: ’cancelled’ });
  t.update(registrationTwoRef, { state: ’cancelled’ });

  return Promise.resolve();
});

return new Promise((resolve, reject) => {
  transaction.then((result) => {
    return resolve();
  }).catch((error) => {
    return reject();
  });
});
```

Obrázek 5.10: Transakce pro změnu stavů dvou registrací na zrušeno.

5.2.5 Struktura

Následuje výpis a popis všech callables funkcí, databázových triggerů, plánovaných funkcí a souborů s pomocnými funkcemi. Pro lepší přehled funkcionalit funkcí jsem zvolil vkládání odpovídajících názvů HTTP dotazovacích metod do názvu funkcí, ikdyž ve skutečnosti komunikace přes HTTP neprobíhá.

- **callable:** Složka obsahující všechny callables funkce.
  - **calendar_get.f.js:** Funkce vrací všechny studentské a lektorské události uživatele.
  - **course_delete.f.js:** Funkce odstraní kurz. Dále odstraní jeho záznamy v dokumentech lektorů tohoto kurzu a semestru, ve kterém byl.
  - **course_patch.f.js:** Funkce aktualizuje kurz. Při změně lokace zvýší příznak usesCount u nové lokace a sníží u té původní. Při změně lektora se do pole taughtCourses dokumentu nového lektora přidá identifikátor aktualizovaného kurzu a v poli původního lektora se tento identifikátor odebere. Dále při změně kapacity nebo odchylky se provede vyhodnocení čekací listiny.

[8]Structured Query Language (SQL) – strukturovaný dotazovací jazyk

- **course__post.f.js:** Funkce vytvoří nový kurz a přidá svůj identifikátor do dokumentů odpovídajících lektorů, semestru a lokace.
- **course-lecture-attendance__post.f.js:** Uloží docházku k dané lekci daného kurzu.
- **course-lecture-video__delete.f.js:** Odstraní odkaz na dané video z lekce a následně toto video vymaže z úložiště.
- **course-lecture-video__post.f.js:** Vloží odkaz na video do dokumentu kurzu dané lekce.
- **lector-from-course__delete.f.js:** Odstraní daného lektora z daného kurzu.
- **lector-invite__post.f.js:** Přidá danému uživateli roli lektora. V případě, že uživatel v systému ještě neexistuje, odešle se na daný email pozvánka na lektorování kurzů. Tato pozvánka obsahuje odkaz na registraci do systému. Po úspěšném zaregistrování pozvaného lektora se mu přiřadí role lektora a zapíší se mu dané kurzy k výuce.
- **lector-role__delete.f.js:** Odebere roli lektora danému uživateli.
- **location__delete.f.js:** Odstraní dokument dané lokace.
- **location__patch.f.js:** Aktualizuje dokument dané lokace a zároveň aktualizuje dokumenty všech kurzů, které probíhají v této lokaci (kvůli denormalizaci).
- **location__post.f.js:** Vytvoří lokaci.
- **payment-info__get.f.js:** Vygeneruje platební údaje k dané registraci.
- **price__get.f.js:** Vrátí vypočítanou cenu na základě daného kurzu a slev.
- **registration__delete.f.js:** Ruší danou registraci a zkontroluje čekací listinu. Po úspěšném zrušení registrace odesílá uživateli informační email o zrušení registrace a jeho typu (zrušení adminem nebo systémem - expirace).
- **registration__post.f.js:** Zpracuje a vytvoří registraci. Vrací stav registrace a informuje uživatele emailem.
- **semester__delete.f.js:** Smaže dokument daného semestru.
- **semester__post.f.js:** Vytvoří nový dokument semestru.
- **semester-activate__post.f.js:** Změní stav daného semestru na aktivní.
- **semester-archive__post.f.js:** Změní stav daného semestru na archivován.
- **semester-copy-courses__post.f.js:** Překopíruje kurzy prvního vybraného semestru do druhého.
- **semester-disable__post.f.js:** Změní stav daného semestru na neaktivní.
- **semester-open__post.f.js:** Otevře daný semestr k registracím.
- **timestamp__get.f.js:** Vrátí aktuální časovou značku serveru.
- **user-add-role__post.f.js:** Přidá danému uživateli vybranou roli.
- **user-avatar__patch.f.js:** Aktualizuje danému uživateli odkaz na jeho profilový obrázek.
- **user-create__post.f.js:** Vytvoří nový účet s daným identifikátorem, emailem a vygenerovaným heslem. Po vytvoření odesílá uživateli vygenerované heslo na daný email. Tato funkce je použita při automatickém vytvoření uživatelského účtu.

  - **user-profile_patch.f.js:** Aktualizuje danému uživateli profil a jedná li se o lektora, který vyučuje nějaký kurz, jsou aktualizovány i tyto kurzy (kvůli denormalizaci).
  - **user-sign-up_post.f.js:** Vytvoří nový účet na základě vyplněné registrace v administrátorském rozhraní.
  - **verification_post.f.js:** Nastaví daný účet jako ověřený. Pakliže má daný účet nějaké přijaté a nezaplacené objednávky, odešle vygenerované platební údaje k zaplacení objednávek.
  - **verification-email_post.f.js:** Odešle ověřovací email na email zvoleného účtu.

- **mailGun:** Složka obsahující funkce pro zasílání jednotlivých druhů emailů.
  - **mailGun.js:** Soubor obsahující funkce pro odesílání emailů.
    * **sendCancelledByAdmin:** Připraví k odeslání email o zrušení objednávky adminem.
    * **sendCancelledByExpiration:** Připraví k odeslání email o zrušení objednávky systémem (nezaplacení objednávky v termínu).
    * **sendPasswordEmail:** Připraví k odeslaní email s vygenerovaným heslem do nového účtu v systému.
    * **sendLectorInvitation:** Připraví k odeslání email s pozvánkou pro nového lektora.
    * **sendRegInWl:** Připraví k odeslání email s informací o zařazení registrace do čekací listiny.
    * **sendShiftedFromWl:** Připraví k odeslání email s informací o uvolnění místa v kurzu a posunutí registrace do kurzu.
    * **sendInvoice:** Připraví k odeslání email s vygenerovanými platebními údaji k zaplacení objednávky.
    * **sendReminderEmail:** Připraví k odeslání email upomínkový email k zaplacení objednávky.
    * **sendVerification:** Vygeneruje odkaz pro ověření účtu a následně připraví k odeslání email s tímto odkazem.
    * **sendPaidEmail:** Připraví k odeslání email s informací o úspěšném zaplacení objednávky.
    * **sendEmail:** Odešle připravený email.

- **schedule:** Složka obsahující plánovaně opakující se callables.
  - **checkFIO.f.js:** Tato funkce se spouští každých pět minut. Komunikuje s FIO bankou a zjišťuje, jestli na bankovní účet přišly nové platby. V případě, že nové platby přišly, vytvoří pro každou tuto platbu dokument v kolekci payments. Do tohoto dokumentu ukládá všechny informace, které API banky poskytuje.
  - **checkRegistrations.f.js:** Tato funkce se spouští každý den v pět hodin ráno. Prochází všechny přijaté registrace, které čekají na zaplacení. Registrace, které nejsou zaplaceny více jak čtyři pracovní dny automaticky ruší a posílá uživatelům daných registrací informační emaily o zrušení kvůli nezaplacení v termínu.

  - **checkSemesters.f.js:** Tato funkce se spouští každý den pět minut po půlnoci. Prochází všechny semestry a mění jim stavy na základě toho, zdali je aktuální datum později než datum konce daného semestru.
  - **registrationsReminder.f.js:** Tato funkce se spouší každý den v půl osmé ráno. Prochází všechny přijaté registrace a odesílá upomínkové emaily klientům, kteří objednávku ještě nezaplatili.

- **triggers:** Složka obsahující všechny triggery
  - **db:** Složka obsahující databázové triggery
    - **payments:** Složka obsahující databázové triggery nad kolekcí payments
      - **onCreate.f.js:** Databázový trigger, který je automaticky spuštěn po vytvoření nového dokumentu v kolekci payments. Tento trigger mění stav dané objednávky na zaplacená v případě, že zaplacená suma odpovídá ceně objednávky.

- **my_functions.js:** Soubor obsahuje funkce, které používají entity napříč celým systémem.

- **my_utils.js:** Soubor obsahuje pomocné funkce.

5.2.6 Zabezpečení

Bez zabezpečení systému by do něj mohl přistupovat kdokoli a to představuje velké bezpečnostní riziko. V tomto systému je nutné zabezpečit všechny Firebase služby, ke kterým se přistupovat. Tyto služby jsou:

- Cloud Firestore
- Cloud Storage
- Cloud Functions pro Firebase

Na zabezpečení databáze a úložiště se používají security rules, které jsou popsány výše. Zabezpečení funkcí (callables) je nutné implementovat v tělech funkcí.

Tabulka 5.1 znázorňuje práva ke čtení a zápisu dokumentů v dané kolekci danou rolí. Tabulka 5.2 znázorňuje práve ke čtení, zápisu a mazání souborů v dané složce danou uživatelskou rolí. Tabulka 5.3 ukazuje kdo a za jakých podmínek může spouštět dané serverové funkce.

Tabulka 5.1: Tabulka znázorňující oprávnění rolí ke čtení / zápisu dokumentů v dané kolekci databáze(* platí pouze pokud tento dokument patří danému uživateli).

	bez role	student	lektor	admin
kurzy	ano / ne	ano / ne	ano / ne	ano / ne
lokace	ne / ne	ne / ne	ne / ne	ano / ne
platby	ne / ne	ano* / ne	ne / ne	ano / ne
registrace	ano / ne	ano / ne	ano / ne	ano / ne
semestry	ano / ne	ano / ne	ano / ne	ano / ne
uživatelé	ne/ ne	ano* / ne	ano* / ne	ano / ne
ostatní	ne / ne	ne / ne	ne / ne	ne / ne

Tabulka 5.2: Tabulka znázorňující oprávnění rolí ke čtení / zápisu / mazání souborů (* platí pouze pokud tento soubor patří danému uživateli).

	bez role	student	lektor	admin
profilové obrázky	ne / ne / ne	ano / ano* / ano*	ano / ano* / ano*	ano / ano* / ano*
výukové videa	ne / ne / ne	ano / ne / ne	ano / ano / ano	ano / ano / ano
ostatní	ne / ne / ne	ne / ne / ne	ne / ne / ne	ne / ne / ne

Tabulka 5.3: Tabulka znázorňující kdo a za jakých podmínek může spouštět jednotlivé serverové funkce.

Serverové funkce (callables)	Podmínky pro nezamítnutí spuštění funkce
calendar_get	přihlášen
course_delete	přihlášen, admin
course_patch	přihlášen, admin
course_post	přihlášen, admin
course-lecture-attendance_post	přihlášen, lektor
course-lecture-video_delete	přihlášen, lektor
course-lecture-video_post	přihlášen, lektor
lector-from-course_delete	přihlášen, admin
lector-invite_post	přihlášen, admin
lector-role_delete	přihlášen, admin
location_delete	přihlášen, admin
location_patch	přihlášen, admin
location_post	přihlášen, admin
price_get	kdokoli
registration_delete	přihlášen, admin
registration_post	kdokoli
semester_delete	přihlášen, admin
semester_post	přihlášen, admin
semester-activate_post	přihlášen, admin
semester-archive_post	přihlášen, admin
semester-copy-courses_post	přihlášen, admin
semester-disable_post	přihlášen, admin
semester-open_post	přihlášen, admin
timestamp_get	kdokoli
user-add-role_post	přihlášen, admin
user-avatar_patch	přihlášen, pokud uživatel upravuje svůj obrázek
user-create_post	kdokoli
user-profile_patch	přihlášen, pokud uživatel upravuje svůj profil
user-sign-up_post	kdokoli
verification_post	kdokoli
verification-email_post	přihlášen, pokud ověřuje svůj email

Kapitola 6

Testování vytvořeného systému

Testování je proces, jenž je součástí zajištění kvality výsledného produktu. Obecným cílem testování je popsat rozdíl mezi skutečnými vlastnostmi produktu/software a danou specifikací. Detailní specifikace chování software však často není dostupná. Stejně tak tomu bylo v případě této práce. Specifikace fungování systému byla pouze ve formě popisu fungování hlavních uživatelských scénářů. Možné méně obvyklé uživatelské scénáře byly popsány a specifikovány až na základě průběžného testování.

6.1 Cíle testování

Software lze testovat s hlediska různých dimenzí kvality. Dle metody FURPS[17] lze testovat následující dimenze kvality:

- **Funkčnost:** Zda software naplňuje požadavky
- **Použitelnost:** Použitelnost systému z pohledu koncového uživatele
- **Spolehlivost:** Četnost a závažnost chyb, schopnost systému chybu ustát
- **Výkon:** Odezva systému na požadavky
- **Udržitelnost:** Schopnost systému být dlouhodobě udržován a spravován

Pro účely nasazení registračního systému v produkčním prostředí jsou prioritními dimenzemi kvality zejména Funkčnost, Použitelnost a Spolehlivost. Výkon systému resp. rychlost jeho odezvy a dlouhodobá udržitelnost jsou pro nás při prvním nasazení méně důležité. Proto jsem se v rámci práce zaměřil především na tyto dvě dimenze kvality.

6.2 Druhy použitého testování

Za účelem testování funkčnosti, použitelnosti a spolehlivosti jsem stavěl hlavně na následujících druzích testů:

- Systémové testování
- Uživatelské testování

6.3 Systémové testování

V rámci systémového testování bylo cílem ověřit funkčnost a spolehlivost systému. Pro tyto účely jsem definoval sadu testovacích scénářů s popisem očekávaného chování systému. Během realizace scénářů jsem narazil na problém jejich velkého rozsahu. Zaměřil jsem se proto na scénáře, které jsou ze zkušeností taneční školy nejvíce pravděpodobné. Jedná se především o následující kategorie scénářů:

- Párová registrace (ověřených či neověřených uživatele/ů)
- Registrace jednotlivce (ověřeného či neověřeného uživatele)
- Opakovaná registrace do kurzu
- Platba objednávky

Na základě výše zmíněných kategorií jsem vytvořil jejich kombinace a spolu s diagramem 4.4 procesu registrace definoval jednotlivé scénáře.

6.4 Uživatelské testování

Cílem uživatelského testování bylo ověřit použitelnost systému a rozhraní z hlediska koncového uživatele. Zde jsem se primárně zaměřil na uživatele - kurzistu. Nejprve jsem s žádostí o testování oslovil pět lidí, kteří prostředí swingových lekcí neznají.

Na osobním setkání jsem pak pozoroval jejich interakce se systémem. Zadání bylo jednoduché: "Představte si, že chcete začít navštěvovat taneční lekce. Vyberte si zde lekci, která vás zaujme a zkuste provést registraci". Dále byli testeři instruováni k takzvanému "přemýšlení nahlas". Z těchto setkání vzešlo několik důležitých zjištění, týkající se uživatelského rozhraní a volby textů, které posléze vedly na úpravu aplikace. Cílem první fáze testování bylo odhalit a vyladit zásadní nedostatky. Následovalo širší testování, kde jsem již nebyl přítomen osobně.

Sestavil jsem základní scénář použití a připravil dotazník pro sběr zpětné vazby na použitelnost systému. Oslovil jsem celkem patnáct lidí s žádostí o zapojení do testování. Úkolem pro testery bylo projít celý scénář a k jednotlivým krokům scénáře poznačit zpětnou vazbu o použitelnosti.

Hlavním zjištěným problémem při testování byla použitelnost na mobilních zařízeních. Testeři reportovali problémy při registraci. Ne vždy také bylo plně srozumitelné, v jakém stavu se provedená registrace nachází. Tyto zjištění následně vedly na úpravy rozhraní aplikace a textace komunikace směrem k uživateli.

Kapitola 7

Závěr

Cílem této práce bylo navrhnout a následně implementovat informační a registrační systém pro spolupracující swingovou taneční školu, která byla nespokojená s jejich dosavadním registračním systémem. Systém umožňuje registraci kurzistů na vypsané taneční kurzy školy, automatickou komunikaci se studenty, zpracování plateb, evidenci docházky studentů a lektorů, nahrávání výukových videí k lekcím a dává vedení školy možnost správy celého systému.

V první etapě bylo nezbytné si se spolupracující školou stanovit funkcionality minimálního produktu, který se nasadí a který se bude nadále rozvíjet. Dále proběhl návrh systému, ve kterém jsem navrhl architekturu systému - modul pro registraci do stávajícího webu školy a administrační rozhraní do nové externí webové stránky. Implementace této architektury, přesněji integrace vytvořeného modulu do stávajícího webu školy, se ukázalo jako nelehký úkol, protože stávající web a nový registrační modul jsou, co se týče použitých technologií, rozdílné. Zvolený postup nasazení modulu popsaný v 5.1.2 se po dalším testování ukázal jako sice funkční, ale ne zcela přívětivý pro uživatele mobilních zařízení, či obecně pomalejších výpočetních strojů. Na základě toho jsem navrhnul další, vylepšené řešení založeném na výše popsaném, lokálním způsobu 5.1.2.

Systém je nasazený na cloudové platformě Firebase, která zajišťuje dobrou škálovatelnost - systém se sám po nečinnosti uspí a naopak při velkém počtu požadavků reaguje rychlým navýšením počtu instancí virtuálních serverů.

Implementace systému byla úspěšně otestována jak lidmi, kteří swingové prostředí neznají, tak těmi, kteří na kurzy pravidelně navštěvují.

7.1 Budoucí vývoj

Výsledek této práce představuje první verzi systému - minimální produkt. Dále tento systém budu rozšiřovat o následující funkcionality:

- Množstevní slevy
- Suplování lektorů
- Předregistrace
- Vícejazyčnost
- SAAS

- Platební brána umožňující více způsobů zaplacení

Literatura

[1] *Fio banka, a.s.* [online]. [cit. 2020-5-07]. Dostupné z: https://www.fio.cz/docs/cz/API_Bankovnictvi.pdf.

[2] *WebChannel* [online]. [cit. 2020-2-07]. Dostupné z: https://google.github.io/closure-library/api/goog.net.WebChannel.html.

[3] Cloudflare, I. *What is BaaS?* [online]. [cit. 2020-15-5]. Dostupné z: https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/.

[4] Google. *Call functions from your app* [online]. [cit. 2020-25-06]. Dostupné z: https://firebase.google.com/docs/functions/callable.

[5] Google. *Cloud Firestore* [online]. [cit. 2020-15-06]. Dostupné z: https://firebase.google.com/docs/firestore.

[6] Google. *Cloud Functions* [online]. [cit. 2020-15-06]. Dostupné z: https://firebase.google.com/docs/functions.

[7] Google. *Cloud Storage* [online]. [cit. 2020-25-06]. Dostupné z: https://firebase.google.com/docs/storage.

[8] Google. *Control Access with Custom Claims and Security Rules* [online]. [cit. 2020-2-07]. Dostupné z: https://firebase.google.com/docs/auth/admin/custom-claims.

[9] Google. *Firebase Authentication* [online]. [cit. 2020-1-07]. Dostupné z: https://firebase.google.com/docs/auth.

[10] Google. *Firebase Hosting* [online]. [cit. 2020-2-07]. Dostupné z: https://firebase.google.com/docs/hosting.

[11] Google. *Firebase Security Rules* [online]. [cit. 2020-2-07]. Dostupné z: https://firebase.google.com/docs/rules.

[12] Google. *Introduction to Angular concepts* [online]. [cit. 2020-10-06]. Dostupné z: https://angular.io/guide/architecture.

[13] Graziotin, D. *Towards a Theory of Aect and Software Developers' Performance* [online]. Bolzano: arxiv.org, leden 2016. Dostupné z: https://arxiv.org/ftp/arxiv/papers/1601/1601.05330.pdf?fbclid=IwAR3SEImKAL6uAOE51YkyQZHmu5IBjSdpV7qksDMeRZH_7YuZObMDMpQHVtg.

[14] Hajian, M. *Progressive Web Apps with Angular*. Apress, 2019. ISBN 978-1-4842-4447-0.

[15] Hutagikar1, V. a Hegde, V. *Analysis of Front-end Frameworks for Web Applications* [online]. irjet.net, duben 2020. Dostupné z: https://www.irjet.net/archives/V7/i4/IRJET-V7I4639.pdf.

[16] Ing. Jaroslav Zendulka doc. a Rudolfová, I. I. *Studijní opora předmětu Databázové systémy* [online]. [cit. 2020-5-07]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIDS-IT%2Ftexts%2FDatabazove_systemy.pdf&cid=13296.

[17] Miguel, J. P., Mauricio, D. a Rodríguez, G. *A Review of Software Quality Models for the Evaluation of Software Products* [online]. arxiv.org, červen 2014. Dostupné z: https://arxiv.org/ftp/arxiv/papers/1412/1412.2977.pdf.

[18] Pano, A., Graziotin, D. a Abrahamsson, P. *Factors and actors leading to the adoption of a JavaScript framework* [online]. arxiv.org, březen 2018. Dostupné z: https://arxiv.org/pdf/1605.04303.pdf.

[19] Terreur, T. *IaaS, PaaS of SaaS? Jij bent de BaaS!* [online]. [cit. 2020-10-5]. Dostupné z: https://www.linkedin.com/pulse/iaas-paas-saas-jij-bent-de-baas-tim-terreur/.

Příloha A

Obsah přiloženého paměťového média

Na přiloženém paměťovém médiu se nachází:

- **zprava** - Složka obsahující zdrojový kód bakalářské práce napsaný v jazyku LaTeX.
- **sb-backend** - Složka obsahující zdrojové soubory backendu.
- **sb-rozhrani-registrace** - Složka obsahující zdrojové soubory webové aplikace pro rozhraní registrace.
- **sb-rozhrani-admin** - Složka obsahující zdrojové soubory webocé aplikace pro rozhraní administrace.
- **informacni-system-pro-swingovou-tanecni-skolu.pdf** - Text této bakalářské práce ve formátu PDF.
- **ukazka.pdf** - Soubor ve formátu PDF s návodem jak si výsledný systém vyzkoušet.